

## AbortProc (3.1)

```
BOOL CALLBACK AbortProc(hdc, error)
HDC hdc;      /* handle of device context */
int error;    /* error value */
```

The AbortProc function is an application-defined callback function that is called when a print job is to be canceled during spooling.

Parameter	Description
hdc	Identifies the device context.
error	Specifies whether an error has occurred. This parameter is zero if no error has occurred; it is <code>SP_OUTOFDISK</code> if Print Manager is currently out of disk space and more disk space will become available if the application waits. If this parameter is <code>SP_OUTOFDISK</code> , the application need not cancel the print job. If it does not cancel the job, it must yield to Print Manager by calling the <a href="#">PeekMessage</a> or <a href="#">GetMessage</a> function.

### Returns

The callback function should return `TRUE` to continue the print job or `FALSE` to cancel the print job.

### Comments

An application installs this callback function by calling the [SetAbortProc](#) function. AbortProc is a placeholder for the application-defined function name. The actual name must be exported by including it in an `EXPORTS` statement in the application's module-definition file.

### See Also

[GetMessage](#), [PeekMessage](#), [SetAbortProc](#)

### CallWndProc (3.1)

```
LRESULT CALLBACK CallWndProc(code, wParam, lParam)
int code;          /* process-message flag */
WPARAM wParam;    /* current-task flag */
LPARAM lParam;    /* address of structure with message data */
```

The `CallWndProc` function is a library-defined callback function that the system calls whenever the `SendMessage` function is called. The system passes the message to the callback function before passing the message to the destination window procedure.

Parameter	Description
code	Specifies whether the callback function should process the message or call the <code>CallNextHookEx</code> function. If the code parameter is less than zero, the callback function should pass the message to <code>CallNextHookEx</code> without further processing.
wParam	Specifies whether the message is sent by the current task. This parameter is nonzero if the message is sent; otherwise, it is NULL.
lParam	Points to a structure that contains details about the message. The following shows the order, type, and description of each member of the structure:
Member	Description
lParam	Contains the lParam parameter of the message.
wParam	Contains the wParam parameter of the message.
uMsg	Specifies the message.
hWnd	Identifies the window that will receive the message.

#### Returns

The callback function should return zero.

#### Comments

The `CallWndProc` callback function can examine or modify the message as necessary. Once the function returns control to the system, the message, with any modifications, is passed on to the window procedure.

This callback function must be in a dynamic-link library.

An application must install the callback function by specifying the `WH_CALLWNDPROC` filter type and the procedure-instance address of the callback function in a call to the `SetWindowsHookEx` function.

`CallWndProc` is a placeholder for the library-defined function name. The actual name must be exported by including it in an `EXPORTS` statement in the library's module-definition file.

#### See Also

[CallNextHookEx](#), [SendMessage](#), [SetWindowsHookEx](#)

### CBTProc (3.1)

```
LRESULT CALLBACK CBTProc(code, wParam, lParam)
int code;          /* CBT hook code          */
WPARAM wParam;    /* depends on the code parameter */
LPARAM lParam;    /* depends on the code parameter */
```

The CBTProc function is a library-defined callback function that the system calls before activating, creating, destroying, minimizing, maximizing, moving, or sizing a window; before completing a system command; before removing a mouse or keyboard event from the system message queue; before setting the input focus; or before synchronizing with the system message queue.

The value returned by the callback function determines whether to allow or prevent one of these operations.

Parameter	Description																		
code	Specifies a computer-based-training (CBT) hook code that identifies the operation about to be carried out, or a value less than zero if the callback function should pass the code, wParam, and lParam parameters to the <u>CallNextHookEx</u> function. The code parameter can be one of the following:																		
	<table><thead><tr><th>Code</th><th>Meaning</th></tr></thead><tbody><tr><td>HCBT_ACTIVATE</td><td>Indicates that the system is about to activate a window.</td></tr><tr><td>HCBT_CLICKSKIPPED</td><td>Indicates that the system has removed a mouse message from the system message queue. A CBT application that must install a journaling playback filter in response to the mouse message should do so when it receives this hook code.</td></tr><tr><td>HCBT_CREATEWND</td><td>Indicates that a window is about to be created. The system calls the callback function before sending the <u>WM_CREATE</u> or <u>WM_NCCREATE</u> message to the window. If the callback function returns TRUE, the system destroys the window--the <u>CreateWindow</u> function returns NULL, but the <u>WM_DESTROY</u> message is not sent to the window. If the callback function returns FALSE, the window is created normally. At the time of the HCBT_CREATEWND notification, the window has been created, but its final size and position may not have been determined, nor has its parent window been established. It is possible to send messages to the newly created window, although the window has not yet received <u>WM_NCCREATE</u> or <u>WM_CREATE</u> messages. It is possible to change the Z-order of the newly created window by modifying the hWndInsertAfter member of the <u>CBT_CREATEWND</u> structure.</td></tr><tr><td>HCBT_DESTROYWND</td><td>Indicates that a window is about to be destroyed.</td></tr><tr><td>HCBT_KEYSKIPPED</td><td>Indicates that the system has removed a keyboard message from the system message queue. A CBT application that must install a journaling playback filter in response to the keyboard message should do so when it receives this hook code.</td></tr><tr><td>HCBT_MINMAX</td><td>Indicates that a window is about to be minimized or maximized.</td></tr><tr><td>HCBT_MOVESIZE</td><td>Indicates that a window is about to be moved or sized.</td></tr><tr><td>HCBT_QS</td><td>Indicates that the system has retrieved a</td></tr></tbody></table>	Code	Meaning	HCBT_ACTIVATE	Indicates that the system is about to activate a window.	HCBT_CLICKSKIPPED	Indicates that the system has removed a mouse message from the system message queue. A CBT application that must install a journaling playback filter in response to the mouse message should do so when it receives this hook code.	HCBT_CREATEWND	Indicates that a window is about to be created. The system calls the callback function before sending the <u>WM_CREATE</u> or <u>WM_NCCREATE</u> message to the window. If the callback function returns TRUE, the system destroys the window--the <u>CreateWindow</u> function returns NULL, but the <u>WM_DESTROY</u> message is not sent to the window. If the callback function returns FALSE, the window is created normally. At the time of the HCBT_CREATEWND notification, the window has been created, but its final size and position may not have been determined, nor has its parent window been established. It is possible to send messages to the newly created window, although the window has not yet received <u>WM_NCCREATE</u> or <u>WM_CREATE</u> messages. It is possible to change the Z-order of the newly created window by modifying the hWndInsertAfter member of the <u>CBT_CREATEWND</u> structure.	HCBT_DESTROYWND	Indicates that a window is about to be destroyed.	HCBT_KEYSKIPPED	Indicates that the system has removed a keyboard message from the system message queue. A CBT application that must install a journaling playback filter in response to the keyboard message should do so when it receives this hook code.	HCBT_MINMAX	Indicates that a window is about to be minimized or maximized.	HCBT_MOVESIZE	Indicates that a window is about to be moved or sized.	HCBT_QS	Indicates that the system has retrieved a
Code	Meaning																		
HCBT_ACTIVATE	Indicates that the system is about to activate a window.																		
HCBT_CLICKSKIPPED	Indicates that the system has removed a mouse message from the system message queue. A CBT application that must install a journaling playback filter in response to the mouse message should do so when it receives this hook code.																		
HCBT_CREATEWND	Indicates that a window is about to be created. The system calls the callback function before sending the <u>WM_CREATE</u> or <u>WM_NCCREATE</u> message to the window. If the callback function returns TRUE, the system destroys the window--the <u>CreateWindow</u> function returns NULL, but the <u>WM_DESTROY</u> message is not sent to the window. If the callback function returns FALSE, the window is created normally. At the time of the HCBT_CREATEWND notification, the window has been created, but its final size and position may not have been determined, nor has its parent window been established. It is possible to send messages to the newly created window, although the window has not yet received <u>WM_NCCREATE</u> or <u>WM_CREATE</u> messages. It is possible to change the Z-order of the newly created window by modifying the hWndInsertAfter member of the <u>CBT_CREATEWND</u> structure.																		
HCBT_DESTROYWND	Indicates that a window is about to be destroyed.																		
HCBT_KEYSKIPPED	Indicates that the system has removed a keyboard message from the system message queue. A CBT application that must install a journaling playback filter in response to the keyboard message should do so when it receives this hook code.																		
HCBT_MINMAX	Indicates that a window is about to be minimized or maximized.																		
HCBT_MOVESIZE	Indicates that a window is about to be moved or sized.																		
HCBT_QS	Indicates that the system has retrieved a																		

WM\_QUEUESYNC message from the system message queue.

HCBT\_SETFOCUS Indicates that a window is about to receive the input focus.

HCBT\_SYSCOMMAND Indicates that a system command is about to be carried out. This allows a CBT application to prevent task switching by hot keys.

wParam This parameter depends on the code parameter. See the following Comments section for details.

lParam This parameter depends on the code parameter. See the following Comments section for details.

#### Returns

For operations corresponding to the following CBT hook codes, the callback function should return zero to allow the operation, or 1 to prevent it:

HCBT\_ACTIVATE  
HCBT\_CREATEWND  
HCBT\_DESTROYWND  
HCBT\_MINMAX  
HCBT\_MOVESIZE  
HCBT\_SYSCOMMAND

The return value is ignored for operations corresponding to the following CBT hook codes:

HCBT\_CLICKSKIPPED  
HCBT\_KEYSKIPPED  
HCBT\_QS

#### Comments

The callback function should not install a playback hook except in the situations described in the preceding list of hook codes.

This callback function must be in a dynamic-link library.

An application must install the callback function by specifying the WH\_CBT filter type and the procedure-instance address of the callback function in a call to the SetWindowsHookEx function.

CBTProc is a placeholder for the library-defined function name. The actual name must be exported by including it in an EXPORTS statement in the library's module-definition file.

The following table describes the wParam and lParam parameters for each HCBT\_ constant.

Constant	wParam	lParam
HCBT_ACTIVATE	Specifies the handle of the window about to be activated.	Specifies a long pointer to a <u>CBTACTIVATESTRUCT</u> structure that contains the handle of the currently active window and specifies whether the activation is changing because of a mouse click.
HCBT_CLICKSKIPPED	Identifies the mouse message removed from the system message queue.	Specifies a long pointer to a <u>MOUSEHOOKSTRUCT</u> structure that contains the hit-test code and the handle of the window for which the mouse message is intended. For a list of hit-test codes, see the description of the <u>WM_NCHITTEST</u> message.
HCBT_CREATEWND	Specifies the handle of the new window.	Specifies a long pointer to a <u>CBT_CREATEWND</u> data structure that contains initialization parameters for the window.
HCBT_DESTROYWND	Specifies the handle of the window about to be destroyed.	This parameter is undefined and should be set to 0L.
HCBT_KEYSKIPPED	Identifies the virtual key code.	Specifies the repeat count, scan code, key-transition code, previous key state, and context code. For more

		information, see the description of the WM_KEYUP or WM_KEYDOWN message.
HCBT_MINMAX	Specifies the handle of the window being minimized or maximized.	The low-order word specifies a show-window value (SW_) that specifies the operation. For a list of show-window values, see the description of the <u>ShowWindow</u> function. The high-order word is undefined.
HCBT_MOVESIZE	Specifies the handle of the window to be moved or sized.	Specifies a long pointer to a <u>RECT</u> structure that contains the coordinates of the window.
HCBT_QS	This parameter is undefined; it should be set to 0.	This parameter is undefined and should be set to 0L.
HCBT_SETFOCUS	Specifies the handle of the window gaining the input focus.	The low-order word specifies the handle of the window losing the input focus. The high-order word is undefined.
HCBT_SYSCOMMAND	Specifies a system-command value (SC_) that specifies the systemcommand. For more information about system command values, see the description of the WM_SYSCOMMAND message.	If wParam is SC_HOTKEY, the low-order word of lParam contains the handle of the window that task switching will bring to the foreground. If wParam is not SC_HOTKEY and a System-menu command is chosen with the mouse, the low-order word of lParam contains the x-coordinate of the cursor and the high-order word contains the y-coordinate. If neither of these conditions is true, lParam is undefined.

See Also

CallNextHookEx, SetWindowsHookEx, CBTACTIVATESTRUCT, CBT\_CREATEWND, RECT

### CPIApplet (3.1)

```
LONG CALLBACK* CPIApplet(hwndCPI, msg, lParam1, lParam2)
HWND hwndCPI;          /* handle of Control Panel window */
UINT msg;              /* message */
/
LPARAM lParam1;
* first message parameter
/
LPARAM lParam2;
* second message parameter
/
```

The CPIApplet function serves as the entry point for a Control Panel dynamic-link library (DLL). This function is supplied by the application.

Parameter	Description
hwndCPI	Identifies the main Control Panel window.
msg	Specifies the message being sent to the DLL.
lParam1	Specifies 32 bits of additional message-dependent information.
lParam2	Specifies 32 bits of additional message-dependent information.

#### Returns

The return value depends on the message.

#### Comments

Use the hwndCPI parameter for dialog boxes or other windows that require a handle of a parent window.

### DdeCallback (3.1)

```
#include <ddeml.h>

HDDEDATA CALLBACK DdeCallback(type, fmt, hconv, hsz1, hsz2, hData, dwData1, dwData2)
UINT type;          /* transaction type */
UINT fmt;           /* clipboard data */
format
/
HCONV hconv;
* handle of conversation
/
HSZ hsz1;
* handle of string
/
HSZ hsz2;
* handle of string
/
HDDEDATA hData;
* handle of global memory object
/
DWORD dwData1;
* transaction-specific data
/
DWORD dwData2;
* transaction-specific data
/
```

The DdeCallback function is an application-defined dynamic data exchange (DDE) callback function that processes DDE transactions sent to the function as a result of DDE Management Library (DDeml) calls by other applications.

Parameter	Description
type	Specifies the type of the current transaction. This parameter consists of a combination of transaction-class flags and transaction-type flags. The following table describes each of the transaction classes and provides a list of the transaction types in each class.

Value	Meaning
XCLASS_BOOL	A DDE callback function should return TRUE or FALSE when it finishes processing a transaction that belongs to this class. Following are the XCLASS_BOOL transaction types:

XCLASS_DATA	<u>X_TYP_ADVSTART</u> <u>X_TYP_CONNECT</u> A DDE callback function should return a DDE data handle, CBR_BLOCK, or NULL when it finishes processing a transaction that belongs to this class. Following are the XCLASS_DATA transaction types:
-------------	---

XCLASS_FLAGS	<u>X_TYP_ADVREQ</u> <u>X_TYP_REQUEST</u> <u>X_TYP_WILDCONNECT</u> A DDE callback function should return DDE_FACK, DDE_FBUSY, or DDE_FNOTPROCESSED when it finishes processing a transaction that belongs to this class. Following are the XCLASS_FLAGS transaction types:
--------------	--

	<u>X_TYP_ADVDATA</u> <u>X_TYP_EXECUTE</u> <u>X_TYP_POKE</u>
--	---

## XCLASS\_NOTIFICATION

The transaction types that belong to this class are for notification purposes only. The return value from the callback function is ignored. Following are the XCLASS\_NOTIFICATION transaction types:

XTYP\_ADVSTOP  
XTYP\_CONNECT\_CONFIRM  
XTYP\_DISCONNECT  
XTYP\_ERROR  
XTYP\_MONITOR  
XTYP\_REGISTER  
XTYP\_XACT\_COMPLETE  
XTYP\_UNREGISTER

fmt	Specifies the format in which data is to be sent or received.
hconv	Identifies the conversation associated with the current transaction.
hsz1	Identifies a string. The meaning of this parameter depends on the type of the current transaction. For more information, see the description of the transaction type.
hsz2	Identifies a string. The meaning of this parameter depends on the type of the current transaction. For more information, see the description of the transaction type.
hData	Identifies DDE data. The meaning of this parameter depends on the type of the current transaction. For more information, see the description of the transaction type.
dwData1	Specifies transaction-specific data. For more information, see the description of the transaction type.
dwData2	Specifies transaction-specific data. For more information, see the description of the transaction type.

### Returns

The return value depends on the transaction class.

### Comments

The callback function is called asynchronously for transactions that do not involve creating or terminating conversations. An application that does not frequently accept incoming messages will have reduced DDE performance because DDEML uses messages to initiate transactions.

An application must register the callback function by specifying its address in a call to the DdeInitialize function. DdeCallback is a placeholder for the application- or library-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition file.

### See Also

DdeEnableCallback, DdeInitialize



## DebugProc (3.1)

```
LRESULT CALLBACK DebugProc(code, wParam, lParam)
int code;          /* hook code */
WPARAM wParam;
* type of hook about to be called
/
LPARAM lParam;
* address of structure with debugging information
/
```

The DebugProc function is a library-defined callback function that the system calls before calling any other filter installed by the SetWindowsHookEx function. The system passes information about the filter about to be called to the DebugProc callback function. The callback function can examine the information and determine whether to allow the filter to be called.

Parameter	Description
code	Specifies the hook code. Currently, HC_ACTION is the only positive valid value. If this parameter is less than zero, the callback function must call the <u>CallNextHookEx</u> function without any further processing.
wParam	Specifies the task handle of the task that installed the filter about to be called.
lParam	Contains a long pointer to a <u>DEBUGHOOKINFO</u> structure.

### Returns

The callback function should return TRUE to prevent the system from calling another filter. Otherwise, the callback function must pass the filter information to the CallNextHookEx function.

### Comments

An application must install this callback function by specifying the WH\_DEBUG filter type and the procedure-instance address of the callback function in a call to the SetWindowsHookEx function.

CallWndProc is a placeholder for the library-defined function name. The actual name must be exported by including it in an EXPORTS statement in the library's module-definition file.

### See Also

CallNextHookEx, SetWindowsHookEx, DEBUGHOOKINFO

## DialogProc (2.x)

```
BOOL CALLBACK DialogProc(HWND hDlg, UINT msg, WPARAM wParam, LPARAM lParam)
/* handle of dialog box */
/* message */
/* first message parameter */
/* second message parameter */
```

The DialogProc function is an application-defined callback function that processes messages sent to a modeless dialog box.

Parameter	Description
hWndDlg	Identifies the dialog box.
msg	Specifies the message.
wParam	Specifies 16 bits of additional message-dependent information.
lParam	Specifies 32 bits of additional message-dependent information.

### Returns

Except in response to the WM\_INITDIALOG message, the dialog box procedure should return nonzero if it processes the message, and zero if it does not. In response to a WM\_INITDIALOG message, the dialog box procedure should return zero if it calls the SetFocus function to set the focus to one of the controls in the dialog box. Otherwise, it should return nonzero, in which case the system will set the focus to the first control in the dialog box that can be given the focus.

### Comments

The dialog box procedure is used only if the dialog box class is used for the dialog box. This is the default class and is used if no explicit class is given in the dialog box template. Although the dialog box procedure is similar to a window procedure, it must not call the DefWindowProc function to process unwanted messages. Unwanted messages are processed internally by the dialog box window procedure.

DialogProc is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition file.

### See Also

CreateDialog, CreateDialogIndirect, CreateDialogIndirectParam, CreateDialogParam, DefWindowProc, SetFocus, WM\_INITDIALOG

### DriverProc (3.1)

```
LRESULT CALLBACK DriverProc(dwDriverIdentifier, hDriver, msg, IParam1, IParam2)
DWORD dwDriverIdentifier;    /* identifies installable driver */
HDRVVR hDriver;              /*
handle of installable driver
/
UINT msg;
* message
/
LPARAM IParam1;
* first message parameter
/
LPARAM IParam2;
* second message parameter
/
```

The DriverProc function processes the specified message.

Parameter	Description																								
dwDriverIdentifier	Specifies an identifier of the installable driver.																								
hDriver	Identifies the installable driver. This parameter is a unique handle that Windows assigns to the driver.																								
msg	Identifies a message that the driver must process. Following are the messages that Windows or an application can send to an installable driver: <table><thead><tr><th>Message</th><th>Description</th></tr></thead><tbody><tr><td>DRV_CLOSE</td><td>Notifies the driver that it should decrement (decrease by one) its usage count and unload the driver if the count is zero.</td></tr><tr><td>DRV_CONFIGURE</td><td>Notifies the driver that it should display a custom-configuration dialog box. (This message should be sent only if the driver returns a nonzero value when the <u>DRV_QUERYCONFIGURE</u> message is processed.)</td></tr><tr><td>DRV_DISABLE</td><td>Notifies the driver that its allocated memory is about to be freed.</td></tr><tr><td>DRV_ENABLE</td><td>Notifies the driver that it has been loaded or reloaded, or that Windows has been enabled.</td></tr><tr><td>DRV_FREE</td><td>Notifies the driver that it will be discarded.</td></tr><tr><td>DRV_INSTALL</td><td>Notifies the driver that it has been successfully installed.</td></tr><tr><td>DRV_LOAD</td><td>Notifies the driver that it has been successfully loaded.</td></tr><tr><td>DRV_OPEN</td><td>Notifies the driver that it is about to be opened.</td></tr><tr><td>DRV_POWER</td><td>Notifies the driver that the device's power source is about to be turned off or turned on.</td></tr><tr><td>DRV_QUERYCONFIGURE</td><td>Determines whether the driver supports the <u>DRV_CONFIGURE</u> message. The message displays a private configuration dialog box.</td></tr><tr><td>DRV_REMOVE</td><td>Notifies the driver that it is about to be removed from the system.</td></tr></tbody></table>	Message	Description	DRV_CLOSE	Notifies the driver that it should decrement (decrease by one) its usage count and unload the driver if the count is zero.	DRV_CONFIGURE	Notifies the driver that it should display a custom-configuration dialog box. (This message should be sent only if the driver returns a nonzero value when the <u>DRV_QUERYCONFIGURE</u> message is processed.)	DRV_DISABLE	Notifies the driver that its allocated memory is about to be freed.	DRV_ENABLE	Notifies the driver that it has been loaded or reloaded, or that Windows has been enabled.	DRV_FREE	Notifies the driver that it will be discarded.	DRV_INSTALL	Notifies the driver that it has been successfully installed.	DRV_LOAD	Notifies the driver that it has been successfully loaded.	DRV_OPEN	Notifies the driver that it is about to be opened.	DRV_POWER	Notifies the driver that the device's power source is about to be turned off or turned on.	DRV_QUERYCONFIGURE	Determines whether the driver supports the <u>DRV_CONFIGURE</u> message. The message displays a private configuration dialog box.	DRV_REMOVE	Notifies the driver that it is about to be removed from the system.
Message	Description																								
DRV_CLOSE	Notifies the driver that it should decrement (decrease by one) its usage count and unload the driver if the count is zero.																								
DRV_CONFIGURE	Notifies the driver that it should display a custom-configuration dialog box. (This message should be sent only if the driver returns a nonzero value when the <u>DRV_QUERYCONFIGURE</u> message is processed.)																								
DRV_DISABLE	Notifies the driver that its allocated memory is about to be freed.																								
DRV_ENABLE	Notifies the driver that it has been loaded or reloaded, or that Windows has been enabled.																								
DRV_FREE	Notifies the driver that it will be discarded.																								
DRV_INSTALL	Notifies the driver that it has been successfully installed.																								
DRV_LOAD	Notifies the driver that it has been successfully loaded.																								
DRV_OPEN	Notifies the driver that it is about to be opened.																								
DRV_POWER	Notifies the driver that the device's power source is about to be turned off or turned on.																								
DRV_QUERYCONFIGURE	Determines whether the driver supports the <u>DRV_CONFIGURE</u> message. The message displays a private configuration dialog box.																								
DRV_REMOVE	Notifies the driver that it is about to be removed from the system.																								
IParam1	Specifies the first message parameter.																								
IParam2	Specifies the second message parameter.																								
Returns	The return value is nonzero if the function is successful. Otherwise, it is zero.																								
Comments																									

The DriverProc function is the main function within a Windows installable driver; it is supplied by the driver developer.

When the msg parameter is DRV\_OPEN, IParam1 is the string following the driver filename from the SYSTEM.INI file and IParam2 is the value given as the IParam parameter in the call to the OpenDriver function.

When the msg parameter is DRV\_CLOSE, IParam1 and IParam2 are the same values as the IParam1 and IParam2 parameters in the call to the CloseDriver function.

See Also

CloseDriver, OpenDriver

## EnumChildProc (2.x)

```
BOOL CALLBACK EnumChildProc(HWND hwnd, LPARAM lParam)
/* handle of child window */
/* application-defined value */
/
```

The EnumChildProc function is an application-defined callback function that receives child window handles as a result of a call to the EnumChildWindows function.

Parameter	Description
hwnd	Identifies a child window of the parent window specified in the <u>EnumChildWindows</u> function.
lParam	Specifies the application-defined value specified in the <u>EnumChildWindows</u> function.

### Returns

The callback function must return nonzero to continue enumeration; to stop enumeration, it must return zero.

### Comments

The callback function can carry out any desired task.

An application must register this callback function by passing its address to the EnumChildWindows function. The EnumChildProc function is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

### See Also

EnumChildWindows

### EnumFontFamProc (3.1)

```
int CALLBACK EnumFontFamProc(lpnlf, lpntm, FontType, lParam)
LOGFONT FAR* lpnlf;          /* address of structure with logical-font data */
TEXTMETRIC FAR* lpntm;
/* address of structure with physical-font data
/
int FontType;
/* type of font
/
LPARAM lParam;
/* address of application-defined data
/
```

The EnumFontFamProc function is an application-defined callback function that retrieves information about available fonts.

Parameter	Description
lpnlf	Points to a <b>NEWLOGFONT</b> structure that contains information about the logical attributes of the font. This structure is locally-defined and is identical to the Windows <b>LOGFONT</b> structure except for two new members. The <b>NEWLOGFONT</b> structure has the following form:

```
struct tagNEWLOGFONT { /* nlf */
    int    lfHeight;
    int    lfWidth;
    int    lfEscapement;
    int    lfOrientation;
    int    lfWeight;
    BYTE   lfItalic;
    BYTE   lfUnderline;
    BYTE   lfStrikeOut;
    BYTE   lfCharSet;
    BYTE   lfOutPrecision;
    BYTE   lfClipPrecision;
    BYTE   lfQuality;
    BYTE   lfPitchAndFamily;
    BYTE   lfFaceName[LF_FACESIZE];
    BYTE   lfFullName[2 * LF_FACESIZE]; /* TrueType only */
/
    BYTE   lfStyle[LF_FACESIZE]; /* TrueType only */
} NEWLOGFONT;
```

The lfFullName and lfStyle members are appended to a **LOGFONT** structure when a TrueType font is enumerated in the EnumFontFamProc function.

The lfFullName member is a character array specifying the full name for the font. This name contains the font name and style name.

The lfStyle member is a character array specifying the style name for the font.

For example, when bold italic Arial® is enumerated, the last three members of the **NEWLOGFONT** structure contain the following strings:

```
lfFaceName = "Arial";
lfFullName = "Arial Bold Italic";
lfStyle = "Bold Italic";
```

See the description of the **LOGFONT** structure for a description of the other members of the structure.

lpntm	Points to a <b>NEWTEXTMETRIC</b> structure that contains information about the physical attributes of the font, if the font is a TrueType font. If the font is not a TrueType font, this parameter points to a <b>TEXTMETRIC</b> structure.
-------	---

FontType	Specifies the type of the font. This parameter can be a combination of the following masks:
----------	---

DEVICE\_FONTTYPE

RASTER\_FONTTYPE  
TRUETYPE\_FONTTYPE

IParam Points to the application-defined data passed by EnumFontFamilies.

Returns

This function must return a nonzero value to continue enumeration; to stop enumeration, it must return zero.

Comments

An application must register this callback function by passing its address to the EnumFontFamilies function. The EnumFontFamProc function is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

The AND (&) operator can be used with the RASTER\_FONTTYPE, DEVICE\_FONTTYPE, and TRUETYPE\_FONTTYPE constants to determine the font type. If the RASTER\_FONTTYPE bit is set, the font is a raster font. If the TRUETYPE\_FONTTYPE bit is set, the font is a TrueType font. If neither bit is set, the font is a vector font. A third mask, DEVICE\_FONTTYPE, is set when a device (for example, a laser printer) supports downloading TrueType fonts or when the font is a device-resident font; it is zero if the device is a display adapter, dot-matrix printer, or other raster device. An application can also use the DEVICE\_FONTTYPE mask to distinguish GDI-supplied raster fonts from device-supplied fonts. GDI can simulate bold, italic, underline, and strikeout attributes for GDI-supplied raster fonts, but not for device-supplied fonts.

See Also

EnumFontFamilies, EnumFonts, LOGFONT, NEWTEXTMETRIC, OUTLINETEXTMETRIC, TEXTMETRIC

### EnumFontsProc (3.1)

```
int CALLBACK EnumFontsProc(lplf, lpntm, FontType, lpData)
LOGFONT FAR* lplf; /* address of logical-font data structure */
NEWTEXTMETRIC FAR* lpntm;
/* address of physical-font data structure
/
int FontType;
/* type of font
/
LPARAM lpData;
/* address of application-defined data
/
```

The EnumFontsProc function is an application-defined callback function that processes font data from the EnumFonts function.

Parameter	Description
lplf	Points to a <u>LOGFONT</u> structure that contains information about the logical attributes of the font.
lpntm	Points to a <u>NEWTEXTMETRIC</u> structure that contains information about the physical attributes of the font, if the font is a TrueType font. If the font is not a TrueType font, this parameter points to a <u>TEXTMETRIC</u> structure. The <u>TEXTMETRIC</u> structure is identical to <u>NEWTEXTMETRIC</u> except that it does not include the last four members.
FontType	Specifies the type of the font. This parameter can be a combination of the following masks:  DEVICE_FONTTYPE RASTER_FONTTYPE TRUETYPE_FONTTYPE
lpData	Points to the application-defined data passed by the <u>EnumFonts</u> function.

#### Returns

This function must return a nonzero value to continue enumeration; to stop enumeration, it must return zero.

#### Comments

An application must register this callback function by passing its address to the EnumFonts function. The EnumFontsProc function is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

The AND (&) operator can be used with the RASTER\_FONTTYPE, DEVICE\_FONTTYPE, and TRUETYPE\_FONTTYPE constants to determine the font type. If the RASTER\_FONTTYPE bit is set, the font is a raster font. If the TRUETYPE\_FONTTYPE bit is set, the font is a TrueType font. If neither bit is set, the font is a vector font. A third mask, DEVICE\_FONTTYPE, is set when a device (for example, a laser printer) supports downloading TrueType fonts or when the font is a device-resident font; it is zero if the device is a display adapter, dot-matrix printer, or other raster device. An application can also use the DEVICE\_FONTTYPE mask to distinguish GDI-supplied raster fonts from device-supplied fonts. GDI can simulate bold, italic, underline, and strikeout attributes for GDI-supplied raster fonts, but not for device-supplied fonts.

#### See Also

EnumFonts, EnumFontFamilies, LOGFONT, NEWTEXTMETRIC, OUTLINETEXTMETRIC, TEXTMETRIC



### EnumMetaFileProc (3.1)

```
int CALLBACK EnumMetaFileProc(hdc, lpht, lpmr, cObj, lParam)
HDC hdc; /* handle of device context */
HANDLETABLE FAR* lpht;
/* address of table of object handles
/
METARECORD FAR* lpmr;
/* address of metafile record
/
int cObj;
/* number of objects in handle table
/
LPARAM lParam;
/* address of application-defined data
/
```

The EnumMetaFileProc function is an application-defined callback function that processes metafile data from the [EnumMetaFile](#) function.

Parameter	Description
hdc	Identifies the special device context that contains the metafile.
lpht	Points to a table of handles associated with the objects (pens, brushes, and so on) in the metafile.
lpmr	Points to a metafile record contained in the metafile.
cObj	Specifies the number of objects with associated handles in the handle table.
lParam	Points to the application-defined data.

#### Returns

The callback function must return a nonzero value to continue enumeration; to stop enumeration, it must return zero.

#### Comments

An application must register this callback function by passing its address to the [EnumMetaFile](#) function.

The EnumMetaFileProc function is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

#### See Also

[EnumMetaFile](#)

### EnumObjectsProc (3.1)

```
int CALLBACK EnumObjectsProc(lpLogObject, lpData)
void FAR* lpLogObject;          /* address of object          */
LPARAM lpData;                 /* address of application-defined data */
/
```

The EnumObjectsProc function is an application-defined callback function that processes object data from the EnumObjects function.

Parameter	Description
lpLogObject	Points to a LOGPEN or LOGBRUSH structure that contains information about the attributes of the object.
lpData	Points to the application-defined data passed by the <u>EnumObjects</u> function.

#### Returns

This function must return a nonzero value to continue enumeration; to stop enumeration, it must return zero.

#### Comments

An application must register this callback function by passing its address to the EnumObjects function. The EnumObjectsProc function is a placeholder for the application-supplied function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

#### Example

The following example retrieves the number of horizontally hatched brushes and fills LOGBRUSH structures with information about each of them:

```
#define MAXBRUSHES 50

GOBJENUMPROC lpProcCallback;
HGLOBAL hglbl;
LPBYTE lpbCountBrush;

lpProcCallback = (GOBJENUMPROC) MakeProcInstance(
    (FARPROC) Callback, hinst);

hglbl = GlobalAlloc(GMEM_FIXED, sizeof(LOGBRUSH)
    * MAXBRUSHES);
lpbCountBrush = (LPBYTE) GlobalLock(hglbl);
*lpbCountBrush = 0;
EnumObjects(hdc, OBJ_BRUSH, lpProcCallback,
    (LPARAM) lpbCountBrush);

FreeProcInstance((FARPROC) lpProcCallback);

int FAR PASCAL Callback(LPLOGBRUSH lpLogBrush, LPBYTE pbData)
{
    /*
    * The pbData parameter contains the number of horizontally
    * hatched brushes; the lpDest parameter is set to follow the
    * byte reserved for pbData and the LOGBRUSH structures that
    * have been filled with brush information.
    */

    LPLOGBRUSH lpDest =
        (LPLOGBRUSH) (pbData + 1 + (*pbData * sizeof(LOGBRUSH)));

    if (lpLogBrush->lbStyle ==
        BS_HATCHED && /* if horiz hatch */
        lpLogBrush->lbHatch == HS_HORIZONTAL) {
        *lpDest++ = *lpLogBrush; /* fills structure with brush info */
    }
}
```

```
    (*pbData)++; /* increments brush count*/  
    if (*pbData >= MAXBRUSHES)  
        return 0;  
    }  
  
    return 1;  
}
```

See Also

[EnumObjects](#), [FreeProcInstance](#), [GlobalAlloc](#), [GlobalLock](#), [MakeProcInstance](#), [LOGBRUSH](#), [LOGPEN](#)

## EnumPropFixedProc (2.x)

```
BOOL CALLBACK EnumPropFixedProc(hwnd, lpsz, hData)
HWND hwnd;          /* handle of window with property */
LPCSTR lpsz;        /* address of
property string or atom
/
HANDLE hData;
* handle data of property data
/
```

The EnumPropFixedProc function is an application-defined callback function that receives a window's property data as a result of a call to the [EnumProps](#) function.

Parameter	Description
hwnd	Identifies the handle of the window that contains the property list.
lpsz	Points to the null-terminated string associated with the property data identified by the hData parameter. The application specified the string and data in a previous call to the <a href="#">SetProp</a> function. If the application passed an atom instead of a string to SetProp, the lpsz parameter contains the atom in the low-order word and zero in the high-order word.
hData	Identifies the property data.

### Returns

The callback function must return TRUE to continue enumeration; it must return FALSE to stop enumeration.

### Comments

This form of the property-enumeration callback function should be used in applications and dynamic-link libraries with fixed data segments and in dynamic libraries with movable data segments that do not contain a stack.

The following restrictions apply to the callback function:

- The callback function must not yield control or do anything that might yield control to other tasks.
- The callback function can call the [RemoveProp](#) function. However, RemoveProp can remove only the property passed to the callback function through the callback function's parameters.
- The callback function should not attempt to add properties.

The EnumPropFixedProc function is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

### See Also

[EnumPropMovableProc](#), [EnumProps](#), [RemoveProp](#), [SetProp](#)

## EnumPropMovableProc (2.x)

```
BOOL CALLBACK EnumPropMovableProc(hwnd, lpsz, hData)
HWND hwnd;          /* handle of window with property */
LPCSTR lpsz;        /* address of
property string or atom
/
HANDLE hData;
* handle of property data
/
```

The EnumPropMovableProc function is an application-defined callback function that receives a window's property data as a result of a call to the [EnumProps](#) function.

Parameter	Description
hwnd	Identifies the handle of the window that contains the property list.
lpsz	Points to the null-terminated string associated with the data identified by the hData parameter. The application specified the string and data in a previous call to the <a href="#">SetProp</a> function. If the application passed an atom instead of a string to SetProp, the lpsz parameter contains the atom.
hData	Identifies the property data.

### Returns

The callback function must return TRUE to continue enumeration; to stop enumeration, it must return FALSE.

### Comments

This form of the property-enumeration callback function should be used in applications with movable data segments and in dynamic libraries whose movable data segments also contain a stack. This form is required since movement of the data will invalidate any long pointer to a variable on the stack, such as the lpsz parameter. The data segment typically moves if the callback function allocates more space in the local heap than is currently available.

The following restrictions apply to the callback function:

- The callback function must not yield control or do anything that might yield control to other tasks.
- The callback function can call the [RemoveProp](#) function. However, RemoveProp can remove only the property passed to the callback function through the callback function's parameters.
- The callback function should not attempt to add properties.

The EnumPropMovableProc function is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

### See Also

[EnumPropFixedProc](#), [EnumProps](#), [RemoveProp](#), [SetProp](#)

## EnumTaskWndProc (2.x)

```
BOOL CALLBACK EnumTaskWndProc(hwnd, lParam)
HWND hwnd;          /* handle of a window */
LPARAM lParam;     /* application-defined
value
/
```

\*

The EnumTaskWndProc function is an application-defined callback function that receives the window handles associated with a task as a result of a call to the EnumTaskWindows function.

Parameter	Description
hwnd	Identifies a window associated with the task specified in the <u>EnumTaskWindows</u> function.
lParam	Specifies the application-defined value specified in the <u>EnumTaskWindows</u> function.

### Returns

The callback function must return TRUE to continue enumeration; to stop enumeration, it must return FALSE.

### Comments

The callback function can carry out any desired task.

The EnumTaskWndProc function is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

### See Also

EnumTaskWindows

## EnumWindowsProc (2.x)

```
BOOL CALLBACK EnumWindowsProc(hwnd, lParam)
HWND hwnd;          /* handle of parent window */
LPARAM lParam;     /* application-defined value */
/
```

The EnumWindowsProc function is an application-defined callback function that receives parent window handles as a result of a call to the EnumWindows function.

Parameter	Description
hwnd	Identifies a parent window.
lParam	Specifies the application-defined value specified in the <u>EnumWindows</u> function.

### Returns

The callback function must return nonzero to continue enumeration; to stop enumeration, it must return zero.

### Comments

The callback function can carry out any desired task.

The EnumWindowsProc function is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

### See Also

EnumWindows

### GetMsgProc (3.1)

```
LRESULT CALLBACK GetMsgProc(code, wParam, lParam)
```

```
int code; /* process-message flag */
```

```
WPARAM wParam;
```

```
/* undefined */
```

```
LPARAM lParam;
```

```
* pointer to MSG structure
```

```
/
```

The `GetMsgProc` function is a library-defined callback function that the system calls whenever the `GetMessage` function has retrieved a message from an application queue. The system passes the retrieved message to the callback function before passing the message to the destination window procedure.

Parameter	Description
code	Specifies whether the callback function should process the message or call the <code>CallNextHookEx</code> function. If this parameter is less than zero, the callback function should pass the message to <code>CallNextHookEx</code> without further processing.
wParam	Specifies a NULL value.
lParam	Points to an <u>MSG</u> structure that contains information about the message.

#### Returns

The callback function should return zero.

#### Comments

The `GetMsgProc` callback function can examine or modify the message as desired. Once the callback function returns control to the system, the `GetMessage` function returns the message, with any modifications, to the application that originally called it. The callback function does not require a return value.

This callback function must be in a dynamic-link library (DLL).

An application must install the callback function by specifying the `WH_GETMESSAGE` filter type and the procedure-instance address of the callback function in a call to the `SetWindowsHookEx` function.

`GetMsgProc` is a placeholder for the library-defined function name. The actual name must be exported by including it in an `EXPORTS` statement in the library's module-definition (`.DEF`) file.

#### See Also

`CallNextHookEx`, `GetMessage`, `SetWindowsHookEx`



## GrayStringProc (2.x)

```
BOOL CALLBACK GrayStringProc(hdc, lpData, cch)
HDC hdc; /* handle of device context */
LPARAM lpData; /* address of
string to be drawn */
int cch;
* length of string to be drawn
/
```

The `GrayStringProc` function is an application-defined callback function that draws a string as a result of a call to the `GrayString` function.

Parameter	Description
<code>hdc</code>	Identifies a device context with a bitmap of at least the width and height specified by the <code>cx</code> and <code>cy</code> parameters passed to the <u><code>GrayString</code></u> function.
<code>lpData</code>	Points to the string to be drawn.
<code>cch</code>	Specifies the length, in characters, of the string.

### Returns

The callback function should return `TRUE` to indicate success. Otherwise it should return `FALSE`.

### Comments

The callback function must draw an image relative to the coordinates (0,0).

`GrayStringProc` is a placeholder for the application-defined function name. The actual name must be exported by including it in an `EXPORTS` statement in the application's module-definition (`.DEF`) file.

### See Also

`GrayString`

### HardwareProc (3.1)

```
LRESULT CALLBACK HardwareProc(code, wParam, lParam)
```

```
int code; /* hook code */
```

```
WPARAM wParam;
```

```
* undefined
```

```
/
```

```
LPARAM lParam;
```

```
* address of structure with event information
```

```
/
```

The `HardwareProc` function is an application-defined callback function that the system calls whenever the application calls the `GetMessage` or `PeekMessage` function and there is a hardware event to process. Mouse events and keyboard events are not processed by this hook.

Parameter	Description
code	Specifies whether the callback function should process the message or call the <code>CallNextHookEx</code> function. If this value is less than zero, the callback function should pass the message to <code>CallNextHookEx</code> without further processing. If this value is <code>HC_NOREMOVE</code> , the application is using the <code>PeekMessage</code> function with the <code>PM_NOREMOVE</code> option, and the message will not be removed from the system queue.
wParam	Specifies a NULL value.
lParam	Points to a <code>HARDWAREHOOKSTRUCT</code> structure.

#### Returns

The callback function should return zero to allow the system to process the message; it should be 1 if the message is to be discarded.

#### Comments

This callback function should not install a playback hook because the function cannot use the `GetMessageExtraInfo` function to get the extra information associated with the message.

The callback function must use the Pascal calling convention and must be declared FAR. An application must install the callback function by specifying the `WH_HARDWARE` filter type and the procedure-instance address of the callback function in a call to the `SetWindowsHookEx` function.

`HardwareProc` is a placeholder for the library-defined function name. The actual name must be exported by including it in an EXPORTS statement in the library's module-definition (.DEF) file.

#### See Also

[CallNextHookEx](#), [GetMessageExtraInfo](#), [SetWindowsHookEx](#), [HARDWAREHOOKSTRUCT](#)

### JournalPlaybackProc (3.1)

```
LRESULT CALLBACK JournalPlaybackProc(code, wParam, lParam)
int code;          /* process-message flag */
WPARAM wParam;
undefined
/
LPARAM lParam;
* address of structure for message
/
```

The `JournalPlaybackProc` function is a library-defined callback function that a library can use to insert mouse and keyboard messages into the system message queue. Typically, a library uses this function to play back a series of mouse and keyboard messages that were recorded earlier by using the `JournalRecordProc` function. Regular mouse and keyboard input is disabled as long as a `JournalPlaybackProc` function is installed.

Parameter	Description
code	Specifies whether the callback function should process the message or call the <code>CallNextHookEx</code> function. If this parameter is less than zero, the callback function should pass the message to <code>CallNextHookEx</code> without further processing.
wParam	Specifies a NULL value.
lParam	Points to an <code>EVENTMSG</code> structure that represents the message being processed by the callback function.

#### Returns

The callback function should return a value that represents the amount of time, in clock ticks, that the system should wait before processing the message. This value can be computed by calculating the difference between the time members of the current and previous input messages. If the function returns zero, the message is processed immediately.

#### Comments

The `JournalPlaybackProc` function should copy an input message to the `lParam` parameter. The message must have been recorded by using a `JournalRecordProc` callback function, which should not modify the message.

Once the function returns control to the system, the message continues to be processed. If the code parameter is `HC_SKIP`, the filter function should prepare to return the next recorded event message on its next call.

This callback function should reside in a dynamic-link library.

An application must install the callback function by specifying the `WH_JOURNALPLAYBACK` filter type and the procedure-instance address of the callback function in a call to the `SetWindowsHookEx` function.

`JournalPlaybackProc` is a placeholder for the library-defined function name. The actual name must be exported by including it in an `EXPORTS` statement in the library's module-definition file.

#### See Also

[CallNextHookEx](#), [JournalRecordProc](#), [SetWindowsHookEx](#), [EVENTMSG](#)

### JournalRecordProc (3.1)

```
LRESULT CALLBACK JournalRecordProc(code, wParam, lParam)
int code;          /* process-message flag */
WPARAM wParam;
undefined
/
LPARAM lParam;
* address of structure for message
/
```

/\*

\*

The `JournalRecordProc` function is a library-defined callback function that records messages that the system removes from the system message queue. Later, a library can use a [JournalPlaybackProc](#) function to play back the messages.

Parameter	Description
code	Specifies whether the callback function should process the message or call the <a href="#">CallNextHookEx</a> function. If this parameter is less than zero, the callback function should pass the message to <a href="#">CallNextHookEx</a> without further processing.
wParam	Specifies a NULL value.
lParam	Points to an <a href="#">MSG</a> structure.

#### Returns

The callback function should return zero.

#### Comments

A `JournalRecordProc` callback function should copy but not modify the messages. After control returns to the system, the message continues to be processed. The callback function does not require a return value.

This callback function must be in a dynamic-link library.

An application must install the callback function by specifying the `WH_JOURNALRECORD` filter type and the procedure-instance address of the callback function in a call to the [SetWindowsHookEx](#) function.

`JournalRecordProc` is a placeholder for the library-defined function name. The actual name must be exported by including it in an `EXPORTS` statement in the library's module-definition file.

#### See Also

[CallNextHookEx](#), [JournalPlaybackProc](#), [SetWindowsHookEx](#)

## KeyboardProc (3.1)

```
LRESULT CALLBACK KeyboardProc(code, wParam, lParam)
int code; /* process-message flag */
WPARAM wParam; /*
virtual-key code
/
LPARAM lParam;
* keyboard-message information
/
```

The KeyboardProc function is a library-defined callback function that the system calls whenever the application calls the [GetMessage](#) or [PeekMessage](#) function and there is a [WM\\_KEYUP](#) or [WM\\_KEYDOWN](#) keyboard message to process.

Parameter	Description
code	Specifies whether the callback function should process the message or call the <a href="#">CallNextHookEx</a> function. If this value is <a href="#">HC_NOREMOVE</a> , the application is using the <a href="#">PeekMessage</a> function with the <a href="#">PM_NOREMOVE</a> option, and the message will not be removed from the system queue. If this value is less than zero, the callback function should pass the message to <a href="#">CallNextHookEx</a> without further processing.
wParam	Specifies the virtual-key code of the given key.
lParam	Specifies the repeat count, scan code, extended key, previous key state, context code, and key-transition state, as shown in the following table. (Bit 0 is the low-order bit):
Bit	Description
0-15	Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.
16-23	Specifies the scan code. The value depends on the original equipment manufacturer (OEM).
24	Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.
25-26	Not used.
27-28	Used internally by Windows.
29	Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.
30	Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.
31	Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.

### Returns

The callback function should return 0 if the message should be processed by the system; it should return 1 if the message should be discarded.

### Comments

This callback function must be in a dynamic-link library.

An application must install the callback function by specifying the [WH\\_KEYBOARD](#) filter type and the procedure-instance address of the callback function in a call to the [SetWindowsHookEx](#) function.

KeyboardProc is a placeholder for the library-defined function name. The actual name must be exported by including it in an EXPORTS statement in the library's module-definition file.

### See Also

[CallNextHookEx](#), [GetMessage](#), [PeekMessage](#), [SetWindowsHookEx](#)

## LibMain (2.x)

```
int CALLBACK LibMain(hinst, wDataSeg, cbHeapSize, lpszCmdLine)
HINSTANCE hinst;          /* handle of library instance */
WORD wDataSeg;           /* library data
segment
/
WORD cbHeapSize;
* default heap size
/
LPSTR lpszCmdLine;
* command-line arguments
/
```

The LibMain function is called by the system to initialize a dynamic-link library (DLL). A DLL must contain the LibMain function if the library is linked with the file LIBENTRY.OBJ.

Parameter	Description
hinst	Identifies the instance of the DLL.
wDataSeg	Specifies the value of the data segment (DS) register.
cbHeapSize	Specifies the size of the heap defined in the module-definition file. (The LibEntry routine in LIBENTRY.OBJ uses this value to initialize the local heap.)
lpszCmdLine	Points to a null-terminated string specifying command-line information. This parameter is rarely used by DLLs.

### Returns

The function should return 1 if it is successful. Otherwise, it should return 0.

### Comments

The LibMain function is called by LibEntry, which is called by Windows when the DLL is loaded. The LibEntry routine is provided in the LIBENTRY.OBJ module. LibEntry initializes the DLL's heap (if a HEAPSIZ value is specified in the DLL's module-definition file) before calling the LibMain function.

### Example

The following example shows a typical LibMain function:

```
int CALLBACK LibMain(HINSTANCE hinst, WORD wDataSeg, WORD cbHeap,
LPSTR lpszCmdLine )
{
    HGLOBAL hgb1ClassStruct;
    LPWNDCLASS lpClassStruct;
    static HINSTANCE hinstLib;

    /* Has the library been initialized yet? */

    if (hinstLib == NULL) {
        hgb1ClassStruct = GlobalAlloc(GHND, sizeof(WNDCLASS));
        if (hgb1ClassStruct != NULL) {
            lpClassStruct = (LPWNDCLASS) GlobalLock(hgb1ClassStruct);
            if (lpClassStruct != NULL) {

                /* Define the class attributes. */

                lpClassStruct->style = CS_HREDRAW | CS_VREDRAW |
CS_DBLCLKS | CS_GLOBALCLASS;
                lpClassStruct->lpfnWndProc = DllWndProc;
                lpClassStruct->cbWndExtra = 0;
                lpClassStruct->hInstance = hinst;
                lpClassStruct->hIcon = NULL;
                lpClassStruct->hCursor = LoadCursor(NULL, IDC_ARROW);
                lpClassStruct->hbrBackground =
                (HBRUSH) (COLOR_WINDOW + 1);
                lpClassStruct->lpszMenuName = NULL;
                lpClassStruct->lpszClassName = "MyClassName";
            }
        }
    }
}
```

```
    hinstLib = (RegisterClass(lpClassStruct)) ?  
    hinst : NULL;  
  
    GlobalUnlock(hgblClassStruct);  
}  
  
    GlobalFree(hgblClassStruct);  
}  
    return (hinstLib ? 1 : 0); /* return 1 = success; 0 = fail */  
}
```

See Also

GlobalAlloc, GlobalFree, GlobalLock, GlobalUnlock, WEP

### LineDDAProc (3.1)

```
void CALLBACK LineDDAProc(xPos, yPos, lpData)
int xPos;          /* x-coordinate of current position */
int yPos;          /* y-coordinate
of current position */
LPARAM lpData;
/* address of application-defined data */
```

The LineDDAProc function is an application-defined callback function that processes coordinates from the LineDDA function.

Parameter	Description
xPos	Specifies the x-coordinate of the current point.
yPos	Specifies the y-coordinate of the current point.
lpData	Points to the application-defined data.

#### Returns

This function does not return a value.

#### Comments

An application must register this function by passing its address to the LineDDA function.

LineDDAProc is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition file.

#### See Also

LineDDA



## LoadProc (2.x)

```
HGLOBAL CALLBACK LoadProc(hglbMem, hinst, hrsrcResInfo)
HGLOBAL hglbMem;          /* handle of object containing resource */
HINSTANCE hinst;          /* handle of application instance */
HRSRC hrsrcResInfo;      /* handle of a resource */
```

The LoadProc function is an application-defined callback function that receives information about a resource to be locked and can process that information as needed.

Parameter	Description
hglbMem	Identifies a memory object that contains a resource. This parameter is NULL if the resource has not yet been loaded.
hinst	Identifies the instance of the module whose executable file contains the resource.
hrsrcResInfo	Identifies the resource. The resource must have been created by using the <a href="#">FindResource</a> function.

### Returns

The return value is a global memory handle for memory that was allocated using the [GMEM\\_DDESHARE](#) flag in the [GlobalAlloc](#) function.

### Comments

If an attempt to lock the memory object identified by the hglbMem parameter fails, this means the resource has been discarded and must be reloaded.

LoadProc is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition file.

### See Also

[FindResource](#), [GlobalAlloc](#), [SetResourceHandler](#)

## MessageProc (3.1)

```
HRESULT CALLBACK MessageProc(code, wParam, lParam)
int code; /* message type */
WPARAM wParam;
undefined
/
LPARAM lParam;
* address of structure with message data
/
```

The MessageProc function is an application- or library-defined callback function that the system calls after a dialog box, message box, or menu has retrieved a message, but before the message is processed. The callback function can process or modify the messages.

Parameter	Description						
code	Specifies the type of message being processed. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MSGF_DIALOGBOX</td><td>Messages inside a dialog box or message box procedure are being processed.</td></tr><tr><td>MSGF_MENU</td><td>Keyboard and mouse messages in a menu are being processed.</td></tr></tbody></table> <p>If the code parameter is less than zero, the callback function must pass the message to <u>CallNextHookEx</u> without further processing and return the value returned by <u>CallNextHookEx</u>.</p>	Value	Meaning	MSGF_DIALOGBOX	Messages inside a dialog box or message box procedure are being processed.	MSGF_MENU	Keyboard and mouse messages in a menu are being processed.
Value	Meaning						
MSGF_DIALOGBOX	Messages inside a dialog box or message box procedure are being processed.						
MSGF_MENU	Keyboard and mouse messages in a menu are being processed.						
wParam	Specifies a NULL value.						
lParam	Points to an <u>MSG</u> structure.						

### Returns

The callback function should return a nonzero value if it processes the message; it should return zero if it does not process the message.

### Comments

The WH\_MSGFILTER filter type is the only task-specific filter. A task may install this filter.

An application must install the callback function by specifying the WH\_MSGFILTER filter type and the procedure-instance address of the callback function in a call to the SetWindowsHookEx function.

MessageProc is a placeholder for the library-defined function name. The actual name must be exported by including it in an EXPORTS statement in the library's module-definition file.

### See Also

CallNextHookEx, SetWindowsHookEx, MSG

### MouseProc (3.1)

```
LRESULT CALLBACK MouseProc(code, wParam, lParam)
int code;          /* process-message flag */
WPARAM wParam;    /* message identifier */
LPARAM lParam;    /* address of MOUSEHOOKSTRUCT structure */
```

The MouseProc function is a library-defined callback function that the system calls whenever an application calls the GetMessage or PeekMessage function and there is a mouse message to be processed.

Parameter	Description
code	Specifies whether the callback function should process the message or call the <u>CallNextHookEx</u> function. If this value is less than zero, the callback function should pass the message to <u>CallNextHookEx</u> without further processing. If this value is <u>HC_NOREMOVE</u> , the application is using a <u>PeekMessage</u> function with the <u>PM_NOREMOVE</u> option, and the message will not be removed from the system queue.
wParam	Specifies the identifier of the mouse message.
lParam	Points to a <u>MOUSEHOOKSTRUCT</u> structure containing information about the mouse.

The callback function should return 0 to allow the system to process the message; it should return 1 to discard the message.

#### Comments

This callback function should not install a JournalPlaybackProc callback function.

An application must install the callback function by specifying the WH\_MOUSE filter type and the procedure-instance address of the callback function in a call to the SetWindowsHookEx function.

MouseProc is a placeholder for the library-defined function name. The actual name must be exported by including it in an EXPORTS statement in the library's module-definition file.

#### See Also

CallNextHookEx, GetMessage, PeekMessage, SetWindowsHookEx

## NotifyProc (2.x)

```
BOOL CALLBACK NotifyProc(hglbl)
HGLOBAL hglbl; /* handle of global memory object */
```

The NotifyProc function is a library-defined callback function that the system calls whenever it is about to discard a global memory object allocated with the GMEM\_NOTIFY flag.

Parameter	Description
hglbl	Identifies the global memory object being discarded.

### Returns

The callback function should return nonzero if the system is to discard the memory object, or zero if it should not.

### Comments

The callback function is not necessarily called in the context of the application that owns the routine. For this reason, the callback function should not assume it is using the stack segment of the application. The callback function should not call any routine that might move memory.

The callback function must be in a fixed code segment of a dynamic-link library.

NotifyProc is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the library's module-definition statement.

### See Also

GlobalNotify

### ShellProc (3.1)

```
LRESULT CALLBACK ShellProc(code, wParam, lParam)
int code;          /* process-message flag */
WPARAM wParam;    /* current-task flag */
/
LPARAM lParam;
* undefined
/
```

The ShellProc function is a library-defined callback function that a shell application can use to receive useful notifications from the system.

Parameter	Description								
code	Specifies a shell-notification code. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>HSHELL_ACTIVATESHELLWINDOW</td><td>The shell application should activate its main window.</td></tr><tr><td>HSHELL_WINDOWCREATED</td><td>A top-level, unowned window was created. The window exists when the system calls a ShellProc function.</td></tr><tr><td>HSHELL_WINDOWDESTROYED</td><td>A top-level, unowned window is about to be destroyed. The window still exists when the system calls a ShellProc function.</td></tr></tbody></table>	Value	Meaning	HSHELL_ACTIVATESHELLWINDOW	The shell application should activate its main window.	HSHELL_WINDOWCREATED	A top-level, unowned window was created. The window exists when the system calls a ShellProc function.	HSHELL_WINDOWDESTROYED	A top-level, unowned window is about to be destroyed. The window still exists when the system calls a ShellProc function.
Value	Meaning								
HSHELL_ACTIVATESHELLWINDOW	The shell application should activate its main window.								
HSHELL_WINDOWCREATED	A top-level, unowned window was created. The window exists when the system calls a ShellProc function.								
HSHELL_WINDOWDESTROYED	A top-level, unowned window is about to be destroyed. The window still exists when the system calls a ShellProc function.								
wParam	Specifies additional information the shell application may need. The interpretation of this parameter depends on the value of the code parameter, as follows: <table><thead><tr><th>code</th><th>wParam</th></tr></thead><tbody><tr><td>HSHELL_ACTIVATESHELLWINDOW</td><td>Not used.</td></tr><tr><td>HSHELL_WINDOWCREATED</td><td>Specifies the handle of the window being created.</td></tr><tr><td>HSHELL_WINDOWDESTROYED</td><td>Specifies the handle of the window being destroyed.</td></tr></tbody></table>	code	wParam	HSHELL_ACTIVATESHELLWINDOW	Not used.	HSHELL_WINDOWCREATED	Specifies the handle of the window being created.	HSHELL_WINDOWDESTROYED	Specifies the handle of the window being destroyed.
code	wParam								
HSHELL_ACTIVATESHELLWINDOW	Not used.								
HSHELL_WINDOWCREATED	Specifies the handle of the window being created.								
HSHELL_WINDOWDESTROYED	Specifies the handle of the window being destroyed.								
lParam	Reserved; not used.								

#### Returns

The return value should be zero.

#### Comments

An application must install this callback function by specifying the `WH_SHELL` filter type and the procedure-instance address of the callback function in a call to the [SetWindowsHook](#) function.

ShellProc is a placeholder for the library-defined function name. The actual name must be exported by including it in an EXPORTS statement in the library's module-definition file.

#### See Also

[DefHookProc](#), [SendMessage](#), [SetWindowsHook](#)

### SysMsgProc (3.1)

```
LRESULT CALLBACK SysMsgProc(code, wParam, lParam)
int code;          /* message type */
WPARAM wParam;    /* undefined */
/
LPARAM lParam;
* pointer to an MSG structure
/
```

The SysMsgProc function is a library-defined callback function that the system calls after a dialog box, message box, or menu has retrieved a message, but before the message is processed. The callback function can process or modify messages for any application in the system.

Parameter	Description						
code	Specifies the type of message being processed. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MSGF_DIALOGBOX</td><td>Messages inside a dialog box or message box procedure are being processed.</td></tr><tr><td>MSGF_MENU</td><td>Keyboard and mouse messages in a menu are being processed.</td></tr></tbody></table> <p>If the code parameter is less than zero, the callback function must pass the message to the <u>CallNextHookEx</u> function without further processing and return the value returned by <u>CallNextHookEx</u>.</p>	Value	Meaning	MSGF_DIALOGBOX	Messages inside a dialog box or message box procedure are being processed.	MSGF_MENU	Keyboard and mouse messages in a menu are being processed.
Value	Meaning						
MSGF_DIALOGBOX	Messages inside a dialog box or message box procedure are being processed.						
MSGF_MENU	Keyboard and mouse messages in a menu are being processed.						
wParam	Must be NULL.						
lParam	Points to the <u>MSG</u> structure to contain the message. The MSG structure has the following form:						

#### Returns

The return value should be nonzero if the function processes the message. Otherwise, it should be zero.

#### Comments

This callback function must be in a dynamic-link library (DLL).

An application must install this callback function by specifying the WH\_SYSMSGFILTER filter type and the procedure-instance address of the callback function in a call to the SetWindowsHookEx function.

SysMsgProc is a placeholder for the library-defined function name. The actual name must be exported by including it in an EXPORTS statement in the library's module-definition file.

#### See Also

CallNextHookEx, MessageBox, SetWindowsHookEx

## TimerProc (2.x)

```
void CALLBACK TimerProc(hwnd, msg, idTimer, dwTime)
HWND hwnd;          /* handle of window for timer messages */
UINT msg;           /* WM_TIMER */
message
/
UINT idTimer;
* timer identifier
/
DWORD dwTime;
* current system time
/
```

The TimerProc function is an application-defined callback function that processes WM\_TIMER messages.

Parameter	Description
hwnd	Identifies the window associated with the timer.
msg	Specifies the <u>WM_TIMER</u> message.
idTimer	Specifies the timer's identifier.
dwTime	Specifies the current system time.

### Returns

This function does not return a value.

### Comments

TimerProc is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition file.

### See Also

KillTimer, SetTimer, WM\_TIMER

## WEP (3.0)

```
int CALLBACK WEP(nExitType)
int nExitType;    /* type of exit */
```

The WEP (Windows exit procedure) callback function performs cleanup for a dynamic-link library (DLL) before the library is unloaded. This function is called by Windows. Although a WEP function was required for every dynamic-link library in previous versions of the Windows operating system, for version 3.1 the WEP function is optional. Most dynamic-link libraries use the WEP function.

Parameter	Description
nExitType	Specifies whether all of Windows is shutting down or only the individual library. This parameter can be either WEP_FREE_DLL or WEP_SYSTEM_EXIT.

### Returns

The return value should be 1 if the function is successful.

### Comments

For Windows version 3.1, WEP is called on the stack of the application that is terminating. This enables WEP to call Windows functions. In Windows version 3.0, however, WEP is called on a KERNEL stack that is too small to process most calls to Windows functions. These calls, including calls to global-memory functions, should be avoided in a WEP function for Windows 3.0. Calls to MS-DOS functions go through a KERNEL intercept and can also overflow the stack in Windows 3.0. There is no general reason to free memory from the global heap in a WEP function, because the kernel frees this kind of memory automatically.

In some low-memory conditions, WEP can be called before the library initialization function is called and before the library's DGROUP data-segment group has been created. A WEP function that relies on the library initialization function should verify that the initialization function has been called. Also, WEP functions that rely on the validity of DGROUP should check for this. The following procedure is recommended for dynamic-link libraries in Windows 3.0; for Windows 3.1, only step 3 is necessary.

- 1 Verify that the data segment is present by using a lar instruction and checking the present bit. This will indicate whether DS has been loaded. (The DS register always contains a valid selector.)
- 2 Set a flag in the data segment when the library initialization is performed. Once the WEP function has verified that the data segment exists, it should test this flag to determine whether initialization has occurred.
- 3 Declare WEP in the EXPORTS section of the module-definition file for the DLL. Following is an example declaration:

```
WEP @1 RESIDENTNAME
```

The keyword RESIDENTNAME makes the name of the function (WEP) resident at all times. (It is not necessary to use the ordinal reference 1.) The name listed in the LIBRARY statement of the module-definition file must be in uppercase letters and must match the name of the DLL file.

Windows calls the WEP function by name when it is ready to remove the DLL. Under low-memory conditions, it is possible for the DLL's nonresident-name table to be discarded from memory. If this occurs, Windows must load the table to determine whether a WEP function was declared for the DLL. Under low-memory conditions, this method could fail, causing a fatal exit. Using the RESIDENTNAME option forces Windows to keep the name entry for WEP in memory whenever the DLL is in use.

In Windows 3.0, WEP must be placed in a fixed code segment. If it is placed instead in a discardable segment, under low-memory conditions Windows must load the WEP segment from disk so that the WEP function can be called before the DLL is discarded. Under certain low-memory conditions, attempting to load the segment containing WEP can cause a fatal exit. When WEP is in a fixed segment, this situation cannot occur. (Because fixed DLL code is also page-locked, you should minimize the amount of fixed code.)

If a DLL is explicitly loaded by calling the LoadLibrary function, its WEP function is called when the DLL is freed by a call to the FreeLibrary function. (The FreeLibrary function should not be called from within a WEP function.) If the DLL is implicitly loaded, WEP is also called, but some debugging applications will indicate that the application has been terminated before WEP is called.

The WEP functions of dependent DLLs can be called in any order. This order depends on the order in which the usage counts for the DLLs reach zero.

See Also



FreeLibrary, LibMain, RegisterClass, UnRegisterClass

## WindowProc (2.x)

```
LRESULT CALLBACK WindowProc(hwnd, msg, wParam, lParam)
```

```
HWND hwnd;          /* handle of window */
```

```
UINT msg;
```

```
/* message
```

```
*/
```

```
WPARAM wParam;
```

```
/* first message parameter
```

```
/
```

```
LPARAM lParam;
```

```
/* second message parameter
```

```
/
```

The WindowProc function is an application-defined callback function that processes messages sent to a window.

Parameter	Description
hwnd	Identifies the window.
msg	Specifies the message.
wParam	Specifies 16 bits of additional message-dependent information.
lParam	Specifies 32 bits of additional message-dependent information.

### Returns

The return value is the result of the message processing. The value depends on the message being processed.

### Comments

The WindowProc name is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition file.

### See Also

DefWindowProc, RegisterClass, WNDCLASS

## WinMain (2.x)

```
int PASCAL WinMain(hinstCurrent, hinstPrevious, lpCmdLine, nCmdShow)
HINSTANCE hinstCurrent;          /* handle of current instance */
HINSTANCE hinstPrevious;
* handle of previous instance
/
LPSTR lpszCmdLine;
* address of command line
/
int nCmdShow;
* show-window type (open/icon)
/
```

The WinMain function is called by the system as the initial entry point for a Windows application.

Parameter	Description																						
hinstCurrent	Identifies the current instance of the application.																						
hinstPrevious	Identifies the previous instance of the application.																						
lpszCmdLine	Points to a null-terminated string specifying the command line for the application.																						
nCmdShow	Specifies how the window is to be shown. This parameter can be one of the following values:																						
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>SW_HIDE</td><td>Hides the window and passes activation to another window.</td></tr><tr><td>SW_MINIMIZE</td><td>Minimizes the specified window and activates the top-level window in the system's list.</td></tr><tr><td>SW_RESTORE</td><td>Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_SHOWNORMAL).</td></tr><tr><td>SW_SHOW</td><td>Activates a window and displays it in its current size and position.</td></tr><tr><td>SW_SHOWMAXIMIZED</td><td>Activates a window and displays it as a maximized window.</td></tr><tr><td>SW_SHOWMINIMIZED</td><td>Activates a window and displays it as an icon.</td></tr><tr><td>SW_SHOWMINNOACTIVE</td><td>Displays a window as an icon. The window that is currently active remains active.</td></tr><tr><td>SW_SHOWNA</td><td>Displays a window in its current state. The window that is currently active remains active.</td></tr><tr><td>SW_SHOWNOACTIVATE</td><td>Displays a window in its most recent size and position. The window that is currently active remains active.</td></tr><tr><td>SW_SHOWNORMAL</td><td>Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_RESTORE).</td></tr></tbody></table>	Value	Meaning	SW_HIDE	Hides the window and passes activation to another window.	SW_MINIMIZE	Minimizes the specified window and activates the top-level window in the system's list.	SW_RESTORE	Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_SHOWNORMAL).	SW_SHOW	Activates a window and displays it in its current size and position.	SW_SHOWMAXIMIZED	Activates a window and displays it as a maximized window.	SW_SHOWMINIMIZED	Activates a window and displays it as an icon.	SW_SHOWMINNOACTIVE	Displays a window as an icon. The window that is currently active remains active.	SW_SHOWNA	Displays a window in its current state. The window that is currently active remains active.	SW_SHOWNOACTIVATE	Displays a window in its most recent size and position. The window that is currently active remains active.	SW_SHOWNORMAL	Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_RESTORE).
Value	Meaning																						
SW_HIDE	Hides the window and passes activation to another window.																						
SW_MINIMIZE	Minimizes the specified window and activates the top-level window in the system's list.																						
SW_RESTORE	Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_SHOWNORMAL).																						
SW_SHOW	Activates a window and displays it in its current size and position.																						
SW_SHOWMAXIMIZED	Activates a window and displays it as a maximized window.																						
SW_SHOWMINIMIZED	Activates a window and displays it as an icon.																						
SW_SHOWMINNOACTIVE	Displays a window as an icon. The window that is currently active remains active.																						
SW_SHOWNA	Displays a window in its current state. The window that is currently active remains active.																						
SW_SHOWNOACTIVATE	Displays a window in its most recent size and position. The window that is currently active remains active.																						
SW_SHOWNORMAL	Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_RESTORE).																						

### Returns

The return value is the return value of the `PostQuitMessage` function if the function is successful. This function returns NULL if it terminates before entering the message loop.

### Comments

The WinMain function calls the instance-initialization function and, if no other instance of the program is running, the application-initialization function. It then performs a message retrieval-and-dispatch loop that is the top-level control structure for the remainder of the application's execution. The loop is terminated when a `WM_QUIT` message is received, at which time this function exits the application instance by returning the value passed by the `PostQuitMessage` function.

### Example

The following example uses the WinMain function to initialize the application (if necessary), initialize the instance, and establish a message loop:

```
int PASCAL WinMain(HINSTANCE hinstCurrent, HINSTANCE hinstPrevious,
    LPSTR lpszCmdLine, int nCmdShow)
{
    MSG msg;

    if (hinstPrevious == NULL) /* other instances? */
    if (!InitApplication(hinstCurrent)) /* shared items*/
    return FALSE; /* initialization failed */

    /* Perform initializations for this instance. */

    if (!InitInstance(hinstCurrent, nCmdShow))
    return FALSE;

    /* Get and dispatch messages until WM_QUIT message. */

    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg); /* translates virtual key codes */
        DispatchMessage(&msg); /* dispatches message to window */
    }
    return (int) msg.wParam; /* return value of PostQuitMessage */
}
```

See Also

DispatchMessage, GetMessage, PostQuitMessage, TranslateMessage

### WordBreakProc (3.1)

```
int CALLBACK WordBreakProc(lpszEditText, ichCurrentWord, cbEditText, action)
LPSTR lpszEditText;          /* address of edit text */
int ichCurrentWord;          /* index of starting point */
/
int cbEditText;              /* length of edit text */
/
int action;                  /* action to take */
/
```

The WordBreakProc function is an application-defined callback function that the system calls whenever a line of text in a multiline edit control must be broken.

Parameter	Description								
lpszEditText	Points to the text of the edit control.								
ichCurrentWord	Specifies an index to a word in the buffer of text that identifies the point at which the function should begin checking for a word break.								
cbEditText	Specifies the number of bytes in the text.								
action	Specifies the action to be taken by the callback function. This parameter can be one of the following values:								
	<table><thead><tr><th>Value</th><th>Action</th></tr></thead><tbody><tr><td>WB_LEFT</td><td>Look for the beginning of a word to the left of the current position.</td></tr><tr><td>WB_RIGHT</td><td>Look for the beginning of a word to the right of the current position.</td></tr><tr><td>WB_ISDELIMITER</td><td>Check whether the character at the current position is a delimiter.</td></tr></tbody></table>	Value	Action	WB_LEFT	Look for the beginning of a word to the left of the current position.	WB_RIGHT	Look for the beginning of a word to the right of the current position.	WB_ISDELIMITER	Check whether the character at the current position is a delimiter.
Value	Action								
WB_LEFT	Look for the beginning of a word to the left of the current position.								
WB_RIGHT	Look for the beginning of a word to the right of the current position.								
WB_ISDELIMITER	Check whether the character at the current position is a delimiter.								

#### Returns

If the action parameter specifies WB\_ISDELIMITER, the return value is non zero (TRUE) if the character at the current position is a delimiter, or zero if it is not. Otherwise, the return value is an index to the beginning of a word in the buffer of text.

#### Comments

A carriage return (CR) followed by a linefeed (LF) must be treated as a single word by the callback function. Two carriage returns followed by a linefeed also must be treated as a single word.

An application must install the callback function by specifying the procedure-instance address of the callback function in a EM\_SETWORDBREAKPROC message.

WordBreakProc is a placeholder for the library-defined function name. The actual name must be exported by including it in an EXPORTS statement in the library's module-definition file.

#### See Also

SendMessage, EM\_SETWORDBREAKPROC

### Callback functions (3.1)

<u>AbortProc</u>	Processes a canceled print job
<u>CallWndProc</u>	Filters messages sent by the SendMessage function
<u>CBTProc</u>	Allows a CBT application to prevent an operation
<u>CPLApplet</u>	Processes messages for a Control Panel DLL
<u>DdeCallback</u>	Processes DDEML transactions
<u>DebugProc</u>	Examines data before it is sent to a hook
<u>DialogProc</u>	Processes messages sent to a modeless dialog box
<u>DriverProc</u>	Processes messages for an installable driver
<u>EnumChildProc</u>	Receives child window handles during enumeration
<u>EnumFontFamProc</u>	Retrieves information about available fonts
<u>EnumFontsProc</u>	Retrieves information about available fonts
<u>EnumMetaFileProc</u>	Processes metafile data
<u>EnumObjectsProc</u>	Processes object data
<u>EnumPropFixedProc</u>	Receives enumerated property data for a window
<u>EnumPropMovableProc</u>	Receives enumerated property data for a window
<u>EnumTaskWndProc</u>	Processes task window handles during enumeration
<u>EnumWindowsProc</u>	Receives parent window handles during enumeration
<u>GetMsgProc</u>	Filters messages retrieved by the GetMessage function
<u>GrayStringProc</u>	Outputs text for the GrayString function
<u>HardwareProc</u>	Filters nonstandard hardware messages
<u>JournalPlaybackProc</u>	Places recorded events into the system queue
<u>JournalRecordProc</u>	Records event messages
<u>KeyboardProc</u>	Filters keyboard messages
<u>LibMain</u>	Initializes a dynamic-link library
<u>LineDDAProc</u>	Processes line data
<u>LoadProc</u>	Receives and processes resource information
<u>MessageProc</u>	Filters dialog box, message box, or menu messages
<u>MouseProc</u>	Filters mouse messages
<u>NotifyProc</u>	Determines whether to discard a global memory object
<u>ShellProc</u>	Receives notifications from the system
<u>SysMsgProc</u>	Filters dialog box, message box, or menu messages
<u>TimerProc</u>	Processes WM_TIMER messages
<u>WEP</u>	Cleans up and exits a dynamic-link library
<u>WindowProc</u>	Processes messages sent to a window
<u>WinMain</u>	Initializes an application and processes message loop
<u>WordBreakProc</u>	Determines line breaks in an edit control

## CommDlgExtendedError (3.1)

```
#include commdlg.h
```

```
DWORD CommDlgExtendedError(void)
```

The `CommDlgExtendedError` function identifies the cause of the most recent error to have occurred during the execution of one of the following common dialog box procedures:

- ChooseColor
- ChooseFont
- FindText
- GetFileTitle
- GetOpenFileName
- GetSaveFileName
- PrintDlg
- ReplaceText

### Returns

The return value is zero if the prior call to a common dialog box procedure was successful. The return value is `CDERR_DIALOGFAILURE` if the dialog box could not be created. Otherwise, the return value is a nonzero integer that identifies an error condition.

### Comments

Following are the possible `CommDlgExtendedError` return values and the meaning of each:

Value	Meaning
<code>CDERR_FINDRESFAILURE</code>	Specifies that the common dialog box procedure failed to find a specified resource.
<code>CDERR_INITIALIZATION</code>	Specifies that the common dialog box procedure failed during initialization. This error often occurs when insufficient memory is available.
<code>CDERR_LOADRESFAILURE</code>	Specifies that the common dialog box procedure failed to load a specified resource.
<code>CDERR_LOCKRESFAILURE</code>	Specifies that the common dialog box procedure failed to lock a specified resource.
<code>CDERR_LOADSTRFAILURE</code>	Specifies that the common dialog box procedure failed to load a specified string.
<code>CDERR_MEMALLOCFailure</code>	Specifies that the common dialog box procedure was unable to allocate memory for internal structures.
<code>CDERR_MEMLOCKFAILURE</code>	Specifies that the common dialog box procedure was unable to lock the memory associated with a handle.
<code>CDERR_NOHINSTANCE</code>	Specifies that the <code>ENABLETEMPLATE</code> flag was set in the <code>Flags</code> member of a structure for the corresponding common dialog box but that the application failed to provide a corresponding instance handle.
<code>CDERR_NOHOOK</code>	Specifies that the <code>ENABLEHOOK</code> flag was set in the <code>Flags</code> member of a structure for the corresponding common dialog box but that the application failed to provide a pointer to a corresponding hook function.
<code>CDERR_NOTEMPLATE</code>	Specifies that the <code>ENABLETEMPLATE</code> flag was set in the <code>Flags</code> member of a structure for the corresponding common dialog box but that the application failed to provide a corresponding template.
<code>CDERR_REGISTERMSGFAIL</code>	Specifies that the <u>RegisterWindowMessage</u> function returned an error value when it was called by the common dialog box procedure.
<code>CDERR_STRUCTSIZE</code>	Specifies as invalid the <code>lStructSize</code> member of a structure for the corresponding common dialog box.
<code>CFERR_NOFONTS</code>	Specifies that no fonts exist.
<code>CFERR_MAXLESSTHANMIN</code>	Specifies that the size specified in the <code>nSizeMax</code> member of the <u>CHOOSEFONT</u> structure is less than the size specified in the <u>nSizeMin</u> member.

FNERR_BUFFERTOOSMALL	Specifies that the buffer for a filename is too small. (This buffer is pointed to by the lpstrFile member of the structure for a common dialog box.)
FNERR_INVALIDFILENAME	Specifies that a filename is invalid.
FNERR_SUBCLASSFAILURE	Specifies that an attempt to subclass a list box failed due to insufficient memory.
FRERR_BUFFERLENGTHZERO	Specifies that a member in a structure for the corresponding common dialog box points to an invalid buffer.
PDERR_CREATEICFAILURE	Specifies that the <u>PrintDlg</u> function failed when it attempted to create an information context.
PDERR_DEFAULTDIFFERENT	Specifies that an application has called the <u>PrintDlg</u> function with the DN_DEFAULTPRN flag set in the wDefault member of the <u>DEVNAMES</u> structure, but the printer described by the other structure members does not match the current default printer. (This happens when an application stores the <u>DEVNAMES</u> structure and the user changes the default printer by using Control Panel.) To use the printer described by the <u>DEVNAMES</u> structure, the application should clear the DN_DEFAULTPRN flag and call the <u>PrintDlg</u> function again. To use the default printer, the application should replace the <u>DEVNAMES</u> structure (and the <u>DEVMODE</u> structure, if one exists) with NULL; this selects the default printer automatically.
PDERR_DNDMMISMATCH	Specifies that the data in the <u>DEVMODE</u> and <u>DEVNAMES</u> structures describes two different printers.
PDERR_GETDEVMODEFAIL	Specifies that the printer driver failed to initialize a <u>DEVMODE</u> structure. (This error value applies only to printer drivers written for Windows versions 3.0 and later.)
PDERR_INITFAILURE	Specifies that the <u>PrintDlg</u> function failed during initialization.
PDERR_LOADDRVFAILURE	Specifies that the <u>PrintDlg</u> function failed to load the device driver for the specified printer.
PDERR_NODEFAULTPRN	Specifies that a default printer does not exist.
PDERR_NODEVICES	Specifies that no printer drivers were found.
PDERR_PARSEFAILURE	Specifies that the <u>PrintDlg</u> function failed to parse the strings in the [devices] section of the WIN.INI file.
PDERR_PRINTERNOTFOUND	Specifies that the [devices] section of the WIN.INI file did not contain an entry for the requested printer.
PDERR_RETDEFFAILURE	Specifies that the PD_RETURNDEFAULT flag was set in the Flags member of the <u>PRINTDLG</u> structure but that either the hDevMode or hDevNames member was nonzero.
PDERR_SETUPFAILURE	Specifies that the <u>PrintDlg</u> function failed to load the required resources.

See Also

ChooseColor, ChooseFont, FindText, GetFileTitle, GetOpenFileName, GetSaveFileName, PrintDlg, ReplaceText



CDERR\_FINDRESFAILURE

Specifies that the common dialog box procedure failed to find a specified resource.

## CDERR\_INITIALIZATION

Specifies that the common dialog box procedure failed during initialization. This error often occurs when insufficient memory is available.

CDERR\_LOADRESFAILURE

Specifies that the common dialog box procedure failed to load a specified resource.

CDERR\_LOCKRESFAILURE

Specifies that the common dialog box procedure failed to lock a specified resource.

CDERR\_LOADSTRFAILURE

Specifies that the common dialog box procedure failed to load a specified string.

CDERR\_MEMALLOCFAILURE

Specifies that the common dialog box procedure was unable to allocate memory for internal structures.

CDERR\_MEMLOCKFAILURE

Specifies that the common dialog box procedure was unable to lock the memory associated with a handle.

## CDERR\_NOINSTANCE

Specifies that the ENABLETEMPLATE flag was set in the Flags member of a structure for the corresponding common dialog box but that the application failed to provide a corresponding instance handle.



## CDERR\_NOHOOK

Specifies that the ENABLEHOOK flag was set in the Flags member of a structure for the corresponding common dialog box but that the application failed to provide a pointer to a corresponding hook function.

## CDERR\_NOTEMPLATE

Specifies that the ENABLETEMPLATE flag was set in the Flags member of a structure for the corresponding common dialog box but that the application failed to provide a corresponding template.

CDERR\_REGISTERMSGFAIL

Specifies that the RegisterWindowMessage function returned an error value when it was called by the common dialog box procedure.

CDERR\_STRUCTSIZE

Specifies as invalid the lStructSize member of a structure for the corresponding common dialog box.

CFERR\_NOFONTS

Specifies that no fonts exist.

CFERR\_MAXLESSTHANMIN

Specifies that the size specified in the nSizeMax member of the CHOOSEFONT structure is less than the size specified in the nSizeMin member.

**FNERR\_BUFFERTOOSMALL**

Specifies that the buffer for a filename is too small. (This buffer is pointed to by the lpstrFile member of the structure for a common dialog box.)

FNERR\_INVALIDFILENAME

Specifies that a filename is invalid.



FNERR\_SUBCLASSFAILURE

Specifies that an attempt to subclass a list box failed due to insufficient memory.

#### FRERR\_BUFFERLENGTHZERO

Specifies that a member in a structure for the corresponding common dialog box points to an invalid buffer.

PDERR\_CREATEICFAILURE

Specifies that the PrintDlg function failed when it attempted to create an information context.

#### PDERR\_DEFAULTDIFFERENT

Specifies that an application has called the PrintDlg function with the DN\_DEFAULTPRN flag set in the wDefault member of the DEVNAMES structure, but the printer described by the other structure members does not match the current default printer. (This happens when an application stores the DEVNAMES structure and the user changes the default printer by using Control Panel.) To use the printer described by the DEVNAMES structure, the application should clear the DN\_DEFAULTPRN flag and call the PrintDlg function again. To use the default printer, the application should replace the DEVNAMES structure (and the DEVMODE structure, if one exists) with NULL; this selects the default printer automatically.

PDERR\_DNDMMISMATCH

Specifies that the data in the DEVMODE and DEVNAMES structures describes two different printers.

PDERR\_GETDEVMODEFAIL

Specifies that the printer driver failed to initialize a DEVMODE structure. (This error value applies only to printer drivers written for Windows versions 3.0 and later.)

PDERR\_INITFAILURE

Specifies that the PrintDlg function failed during initialization.

PDERR\_LOADDRVFAILURE

Specifies that the PrintDlg function failed to load the device driver for the specified printer.



PDERR\_NODEFAULTPRN

Specifies that a default printer does not exist.

PDERR\_NODEVICES

Specifies that no printer drivers were found.

PDERR\_PARSEFAILURE

Specifies that the PrintDlg function failed to parse the strings in the [devices] section of the WIN.INI file.

PDERR\_PRINTERNOTFOUND

Specifies that the [devices] section of the WIN.INI file did not contain an entry for the requested printer.

PDERR\_RETDEFFAILURE

Specifies that the PD\_RETURNDEFAULT flag was set in the Flags member of the PRINTDLG structure but that either the hDevMode or hDevNames member was nonzero.

PDERR\_SETUPFAILURE

Specifies that the PrintDlg function failed to load the required resources.

## ChooseColor (3.1)

#include commdlg.h

```
BOOL ChooseColor(lpcc)
CHOOSECOLOR FAR* lpcc;          /* address of structure with initialization data */
```

The ChooseColor function creates a system-defined dialog box from which the user can select a color.

Parameter	Description
lpcc	Points to a <u>CHOOSECOLOR</u> structure that initially contains information necessary to initialize the dialog box. When the ChooseColor function returns, this structure contains information about the user's color selection.

### Returns

The return value is nonzero if the function is successful. It is zero if an error occurs, if the user chooses the Cancel button, or if the user chooses the Close command on the System menu (often called the Control menu) to close the dialog box.

### Errors

Use the CommDlgExtendedError function to retrieve the error value, which may be one of the following:

```
CDERR_FINDRESFAILURE
CDERR_INITIALIZATION
CDERR_LOCKRESFAILURE
CDERR_LOADRESFAILURE
CDERR_LOADSTRFAILURE
CDERR_MEMALLOCFailure
CDERR_MEMLOCKFAILURE
CDERR_NOHINSTANCE
CDERR_NOHOOK
CDERR_NOTEMPLATE
CDERR_STRUCTSIZE
```

### Comments

The dialog box does not support color palettes. The color choices offered by the dialog box are limited to the system colors and dithered versions of those colors.

If the hook function (to which the lpfnHook member of the CHOOSECOLOR structure points) processes the WM\_CTLCOLOR message, this function must return a handle for the brush that should be used to paint the control background.

### Example

The following example initializes a CHOOSECOLOR structure and then creates a color-selection dialog box:

```
/* Color variables */

CHOOSECOLOR cc;
COLORREF clr;
COLORREF aclrCust[16];
int i;

/* Set the custom color controls to white. */

for (i = 0; i < 16; i++)
    aclrCust[i] = RGB(255, 255, 255);

/* Initialize clr to black. */

clr = RGB(0, 0, 0);

/* Set all structure fields to zero. */

memset(&cc, 0, sizeof(CHOOSECOLOR));
```

```
/* Initialize the necessary CHOOSECOLOR members. */  
  
cc.lStructSize = sizeof(CHOOSECOLOR);  
cc.hwndOwner = hwnd;  
cc.rgbResult = clr;  
cc.lpCustColors = aclrCust;  
cc.Flags = CC_PREVENTFULLOPEN;  
  
if (ChooseColor(&cc))  
    .  
    . /* Use cc.rgbResult to select the user-requested color. */  
    .
```

See Also  
CHOOSECOLOR

The following shows how the dialog box normally appears:



## ChooseFont function (3.1)

#include commdlg.h

```
BOOL ChooseFont(lpcef)
CHOOSEFONT FAR*lpcef;          /* address of structure with initialization data */
```

The ChooseFont function creates a system-defined dialog box from which the user can select a font, a font style (such as bold or italic), a point size, an effect (such as strikethrough or underline), and a color.

Parameter	Description
lpcef	Points to a <b>CHOOSEFONT</b> structure that initially contains information necessary to initialize the dialog box. When the ChooseFont function returns, this structure contains information about the user's font selection.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Errors

Use the [CommDlgExtendedError](#) function to retrieve the error value, which may be one of the following:

```
CDERR_FINDRESFAILURE
CDERR_INITIALIZATION
CDERR_LOCKRESFAILURE
CDERR_LOADRESFAILURE
CDERR_LOADSTRFAILURE
CDERR_MEMALLOCFailure
CDERR_MEMLOCKFAILURE
CDERR_NOHINSTANCE
CDERR_NOHOOK
CDERR_NOTEMPLATE
CDERR_STRUCTSIZE
CFERR_MAXLESSTHANMIN
CFERR_NOFONTS
```

### Example

The following example initializes a **CHOOSEFONT** structure and then displays a font dialog box:

```
LOGFONT lf;
CHOOSEFONT cf;

/* Set all structure fields to zero. */

memset(&cf, 0, sizeof(CHOOSEFONT));

cf.lStructSize = sizeof(CHOOSEFONT);
cf.hwndOwner = hwnd;
cf.lpLogFont = &lf;
cf.Flags = CF_SCREENFONTS | CF_EFFECTS;
cf.rgbColors = RGB(0, 255, 255); /* light blue */
cf.nFontType = SCREEN_FONTTYPE;

ChooseFont(&cf);
```

See Also

## CHOOSEFONT

The following shows how the dialog box normally appears:

## FindText (3.1)

#include commdlg.h

HWND FindText(lpfr)

FINDREPLACE FAR\* lpfr; /\* address of structure with initialization data \*/

The FindText function creates a system-defined modeless dialog box that makes it possible for the user to find text within a document. The application must perform the search operation.

Parameter	Description
lpfr	Points to a <u>FINDREPLACE</u> structure that contains information used to initialize the dialog box. When the user makes a selection in the dialog box, the system fills this structure with information about the user's selection and then sends a message to the application. This message contains a pointer to the FINDREPLACE structure.

### Returns

The return value is the window handle of the dialog box if the function is successful. Otherwise, it is NULL. An application can use this window handle to communicate with or to close the dialog box.

### Errors

Use the CommDlgExtendedError function to retrieve the error value, which may be one of the following values:

CDERR\_FINDRESFAILURE  
CDERR\_INITIALIZATION  
CDERR\_LOCKRESFAILURE  
CDERR\_LOADRESFAILURE  
CDERR\_LOADSTRFAILURE  
CDERR\_MEMALLOCFAILURE  
CDERR\_MEMLOCKFAILURE  
CDERR\_NOHINSTANCE  
CDERR\_NOHOOK  
CDERR\_NOTEMPLATE  
CDERR\_STRUCTSIZE  
FRERR\_BUFFERLENGTHZERO

### Comments

The dialog box procedure for the Find dialog box passes user requests to the application through special messages. The lParam parameter of each of these messages contains a pointer to a FINDREPLACE structure. The procedure sends the messages to the window identified by the hwndOwner member of the FINDREPLACE structure. An application can register the identifier for these messages by specifying the "commdlg\_FindReplace" string in a call to the RegisterWindowMessage function.

For the TAB key to function correctly, any application that calls the FindText function must also call the IsDialogMessage function in its main message loop. (The IsDialogMessage function returns a value that indicates whether messages are intended for the Find dialog box.)

If the hook function (to which the lpfnHook member of the FINDREPLACE structure points) processes the WM\_CTLCOLOR message, this function must return a handle of the brush that should be used to paint the control background.

### Example

The following example initializes a FINDREPLACE structure and calls the FindText function to display the Find dialog box:

```
FINDREPLACE fr;
```

```

/* Set all structure fields to zero. */
memset(&fr, 0, sizeof(FINDREPLACE));

fr.lStructSize = sizeof(FINDREPLACE);
fr.hwndOwner = hwnd;
fr.lpstrFindWhat = szFindWhat;
fr.wFindWhatLen = sizeof(szFindWhat);

hDlg = FindText(&fr);

break;

```

In addition to initializing the members of the **FINDREPLACE** structure and calling the FindText function, an application must register the special **FINDMSGSTRING** message and process messages from the dialog box.

The following example registers the message by using the RegisterWindowMessage function:

```

UINT uFindReplaceMsg;

/* Register the FindReplace message. */

uFindReplaceMsg = RegisterWindowMessage(FINDMSGSTRING);

```

After the application registers the **FINDMSGSTRING** message, it can process messages by using the RegisterWindowMessage return value. The following example processes messages for the Find dialog box and then calls its own SearchFile function to locate the string of text. If the user is closing the dialog box (that is, if the Flags member of the **FINDREPLACE** structure is **FR\_DIALOGTERM**), the handle is invalidated and the procedure returns zero.

```

LRESULT CALLBACK MainWndProc(HWND hwnd, UINT msg, WPARAM wParam,
LPARAM lParam)
{
    FINDREPLACE FAR* lpfr;

    if (msg == uFindReplaceMsg) {
        lpfr = (FINDREPLACE FAR*) lParam;
        if (lpfr->Flags & FR_DIALOGTERM) {
            hDlg = NULL;
            return 0;
        }
        SearchFile(BOOL (lpfr->Flags & FR_DOWN),
(BOOL (lpfr->Flags & FR_MATCHCASE));
        return 0;
    }
}

```

See Also

IsDialogMessage, RegisterWindowMessage, ReplaceText, FINDREPLACE

The following shows how the find dialog box appears:

### GetFileTitle (3.1)

```
#include commdlg.h
```

```
int GetFileTitle(lpszFile, lpszTitle, cbBuf)
```

```
LPCSTR lpszFile;      /* pointer to filename (including drive and directory) */
```

```
LPSTR lpszTitle;
```

```
* address of buffer that receives filename
```

```
/
```

```
UINT cbBuf;
```

```
* length of buffer
```

```
/
```

The GetFileTitle function returns the title of the file identified by the lpszFile parameter.

Parameter	Description
lpszFile	Points to the name and location of an MS-DOS file.
lpszTitle	Points to a buffer into which the function is to copy the name of the file.
cbBuf	Specifies the length, in bytes, of the buffer to which the lpszTitle parameter points.

#### Returns

The return value is zero if the function is successful. The return value is a negative number if the filename is invalid. The return value is a positive integer that specifies the required buffer size, in bytes, if the buffer to which the lpszTitle parameter points is too small.

#### Comments

The function returns an error value if the buffer pointed to by the lpszFile parameter contains any of the following:

- An empty string
- A string containing a wildcard (\*), opening bracket ([), or closing bracket (])
- A string that ends with a colon (:), slash mark (/), or backslash (\)
- A string whose length exceeded the length of the buffer
- An invalid character (for example, a space or unprintable character).

The required buffer size includes the terminating null character.

## GetOpenFileName (3.1)

#include commdlg.h

```
BOOL GetOpenFileName(lpfn)
OPENFILENAME FAR* lpfn;          /* address of initialization data structure */
```

The `GetOpenFileName` function creates a system-defined dialog box that makes it possible for the user to select a file to open.

Parameter	Description
<code>lpfn</code>	Points to an <code>OPENFILENAME</code> structure that contains information used to initialize the dialog box. When the <code>GetOpenFileName</code> function returns, this structure contains information about the user's file selection.

### Returns

The return value is nonzero if the user selects a file to open. It is zero if an error occurs, if the user chooses the Cancel button, if the user chooses the Close command on the System menu to close the dialog box, or if the buffer identified by the `lpstrFile` member of the `OPENFILENAME` structure is too small to contain the string that specifies the selected file.

### Errors

The `CommDlgExtendedError` function retrieves the error value, which may be one of the following values:

```
CDERR_FINDRESFAILURE
CDERR_INITIALIZATION
CDERR_LOCKRESFAILURE
CDERR_LOADRESFAILURE
CDERR_LOADSTRFAILURE
CDERR_MEMALLOCFailure
CDERR_MEMLOCKFAILURE
CDERR_NOINSTANCE
CDERR_NOHOOK
CDERR_NOTEMPLATE
CDERR_STRUCTSIZE
FNERR_BUFFERTOOSMALL
FNERR_INVALIDFILENAME
FNERR_SUBCLASSFAILURE
```

### Comments

If the hook function (to which the `lpfnHook` member of the `OPENFILENAME` structure points) processes the `WM_CTLCOLOR` message, this function must return a handle of the brush that should be used to paint the control background.

### Example

The following example copies file-filter strings into a buffer, initializes an `OPENFILENAME` structure, and then creates an Open dialog box.

The file-filter strings are stored in the resource file in the following form:

```
STRINGTABLE
BEGIN
    IDS_FILTERSTRING    "Write Files(*.WRI)|*.wri|Word Files(*.DOC)|*.
doc|"
END
```

The replaceable character at the end of the string is used to break the entire string into separate strings, while still guaranteeing that all the strings are contiguous in memory.

```
OPENFILENAME ofn;
char szDirName[256];
char szFile[256], szFileTitle[256];
UINT i, cbString;
char chReplace; /* string separator for szFilter */
char szFilter[256];
HFILE hf;
```

```

/* Get the system directory name, and store in szDirName */
GetSystemDirectory(szDirName, sizeof(szDirName));
szFile[0] = '\\0';

if ((cbString = LoadString(hinst, IDS_FILTERSTRING,
    szFilter, sizeof(szFilter))) == 0) {
    ErrorHandler();
    return 0L;
}
chReplace = szFilter[cbString - 1]; /* retrieve wildcard */

for (i = 0; szFilter[i] != '\\0'; i++) {
    if (szFilter[i] == chReplace)
        szFilter[i] = '\\0';
}

/* Set all structure members to zero. */
memset(&ofn, 0, sizeof(OPENFILENAME));

ofn.lStructSize = sizeof(OPENFILENAME);
ofn.hwndOwner = hwnd;
ofn.lpstrFilter = szFilter;
ofn.nFilterIndex = 1;
ofn.lpstrFile = szFile;
ofn.nMaxFile = sizeof(szFile);
ofn.lpstrFileTitle = szFileTitle;
ofn.nMaxFileTitle = sizeof(szFileTitle);
ofn.lpstrInitialDir = szDirName;
ofn.Flags = OFN_SHOWHELP | OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;

if (GetOpenFileName(&ofn)) {
    hf = _lopen(ofn.lpstrFile, OF_READ);
    .
    . /* Perform file operations. */
    .
}
else
    ErrorHandler();

```

See Also

GetSaveFileName, OPENFILENAME

The following shows how the open dialog box normally appears:

## GetSaveFileName (3.1)

```
#include commdlg.h
```

```
BOOL GetSaveFileName(lpofn)
OPENFILENAME FAR* lpofn;          /* address of initialization data */
```

The `GetSaveFileName` function creates a system-defined dialog box that makes it possible for the user to select a file to save.

Parameter	Description
<code>lpofn</code>	Points to an <code>OPENFILENAME</code> structure that contains information used to initialize the dialog box. When the <code>GetSaveFileName</code> function returns, this structure contains information about the user's file selection.

### Returns

The return value is nonzero if the user selects a file to save. It is zero if an error occurs, if the user clicks the Cancel button, if the user chooses the Close command on the System menu to close the dialog box, or if the buffer identified by the `lpstrFile` member of the `OPENFILENAME` structure is too small to contain the string that specifies the selected file.

### Errors

The `CommDlgExtendedError` retrieves the error value, which may be one of the following values:

```
CDERR_FINDRESFAILURE
CDERR_INITIALIZATION
CDERR_LOCKRESFAILURE
CDERR_LOADRESFAILURE
CDERR_LOADSTRFAILURE
CDERR_MEMALLOCFailure
CDERR_MEMLOCKFAILURE
CDERR_NOHINSTANCE
CDERR_NOHOOK
CDERR_NOTEMPLATE
CDERR_STRUCTSIZE
FNERR_BUFFERTOOSMALL
FNERR_INVALIDFILENAME
FNERR_SUBCLASSFAILURE
```

### Comments

If the hook function (to which the `lpfnHook` member of the `OPENFILENAME` structure points) processes the `WM_CTLCOLOR` message, this function must return a handle for the brush that should be used to paint the control background.

### Example

The following example copies file-filter strings (filename extensions) into a buffer, initializes an `OPENFILENAME` structure, and then creates a Save As dialog box.

The file-filter strings are stored in the resource file in the following form:

```
STRINGTABLE
BEGIN
    IDS_FILTERSTRING    "Write Files(*.WRI)|*.wri|Word Files(*.DOC)|*.
doc|"
END
```

The replaceable character at the end of the string is used to break the entire string into separate strings, while still guaranteeing that all the strings are contiguous in memory.



```

OPENFILENAME ofn;
char szDirName[256];
char szFile[256], szFileTitle[256];
UINT i, cbString;
char chReplace; /* string separator for szFilter */
char szFilter[256];
HFILE hf;

/*
 * Retrieve the system directory name, and store it in
 * szDirName.
 */

GetSystemDirectory(szDirName, sizeof(szDirName));

if ((cbString = LoadString(hinst, IDS_FILTERSTRING,
    szFilter, sizeof(szFilter))) == 0) {
    ErrorHandler();
    return 0;
}

chReplace = szFilter[cbString - 1]; /* retrieve wildcard */

for (i = 0; szFilter[i] != '\0'; i++) {
    if (szFilter[i] == chReplace)
        szFilter[i] = '\0';
}

/* Set all structure members to zero. */

memset(&ofn, 0, sizeof(OPENFILENAME));

/* Initialize the OPENFILENAME members. */

szFile[0] = '\0';

ofn.lStructSize = sizeof(OPENFILENAME);
ofn.hwndOwner = hwnd;
ofn.lpstrFilter = szFilter;
ofn.lpstrFile = szFile;
ofn.nMaxFile = sizeof(szFile);
ofn.lpstrFileTitle = szFileTitle;
ofn.nMaxFileTitle = sizeof(szFileTitle);
ofn.lpstrInitialDir = szDirName;
ofn.Flags = OFN_SHOWHELP | OFN_OVERWRITEPROMPT;

if (GetSaveFileName(&ofn)) {
    .
    . /* Perform file operations. */
    .
}
else
    ErrorHandler();

```

See Also

[GetOpenFileName](#), [OPENFILENAME](#)

The following shows how the save dialog box appears:

## PrintDlg function (3.1)

```
#include commdlg.h
```

```
BOOL PrintDlg(lppd)  
PRINTDLG FAR* lppd;          /* address of structure with initialization data */
```

The PrintDlg function displays a Print dialog box or a Print Setup dialog box. The Print dialog box makes it possible for the user to specify the properties of a particular print job. The Print Setup dialog box makes it possible for the user to select additional job properties and configure the printer.

Parameter	Description
lppd	Points to a PRINTDLG structure that contains information used to initialize the dialog box. When the PrintDlg function returns, this structure contains information about the user's selections.

### Returns

The return value is nonzero if the function successfully configures the printer. The return value is zero if an error occurs, if the user chooses the Cancel button, or if the user chooses the Close command on the System menu to close the dialog box. (The return value is also zero if the user chooses the Setup button to display the Print Setup dialog box, chooses the OK button in the Print Setup dialog box, and then chooses the Cancel button in the Print dialog box.)

### Errors

Use the [CommDlgExtendedError](#) function to retrieve the error value, which may be one of the following:

CDERR_FINDRESFAILURE	PDERR_CREATEICFAILURE
CDERR_INITIALIZATION	PDERR_DEFAULTDIFFERENT
CDERR_LOADRESFAILURE	PDERR_DNDMMISMATCH
CDERR_LOADSTRFAILURE	PDERR_GETDEVMODEFAIL
CDERR_LOCKRESFAILURE	PDERR_INITFAILURE
CDERR_MEMALLOCFailure	PDERR_LOADDRVFAILURE
CDERR_MEMLOCKFAILURE	PDERR_NODEFAULTPRN
CDERR_NOHINSTANCE	PDERR_NODEVICES
CDERR_NOHOOK	PDERR_PARSEFAILURE
CDERR_NOTEMPLATE	PDERR_PRINTERNOTFOUND
CDERR_STRUCTSIZE	PDERR_RETDEFFAILURE
	PDERR_SETUPFAILURE

### Example

The following example initializes the PRINTDLG structure, calls the PrintDlg function to display the Print dialog box, and prints a sample page of text if the return value is nonzero:

```
PRINTDLG pd;  
  
/* Set all structure members to zero. */  
  
memset(&pd, 0, sizeof(PRINTDLG));  
  
/* Initialize the necessary PRINTDLG structure members. */  
  
pd.lStructSize = sizeof(PRINTDLG);  
pd.hwndOwner = hwnd;  
pd.Flags = PD_RETURNDC;
```

```

/* Print a test page if successful */
if (PrintDlg(&pd) != 0) {
    Escape(pd.hDC, STARTDOC, 8, "Test-Doc", NULL);

    /* Print text and rectangle */

    TextOut(pd.hDC, 50, 50, "Common Dialog Test Page", 23);
    Rectangle(pd.hDC, 50, 90, 625, 105);
    Escape(pd.hDC, NEWFRAME, 0, NULL, NULL);
    Escape(pd.hDC, ENDDOC, 0, NULL, NULL);
    DeleteDC(pd.hDC);
    if (pd.hDevMode != NULL)
        GlobalFree(pd.hDevMode);
    if (pd.hDevNames != NULL)
        GlobalFree(pd.hDevNames);
}
else
    ErrorHandler();

```

See Also

PRINTDLG

The following shows how the print dialog box normally appears:

## ReplaceText (3.1)

#include commdlg.h

```
HWND ReplaceText(lpfr)
FINDREPLACE FAR* lpfr;          /* address of structure with initialization data */
```

The ReplaceText function creates a system-defined modeless dialog box that makes it possible for the user to find and replace text within a document. The application must perform the actual find and replace operations.

Parameter	Description
lpfr	Points to a <u>FINDREPLACE</u> structure that contains information used to initialize the dialog box. When the user makes a selection in the dialog box, the system fills this structure with information about the user's selection and then sends a message to the application. This message contains a pointer to the FINDREPLACE structure.

### Returns

The return value is the window handle of the dialog box, or it is NULL if an error occurs. An application can use this handle to communicate with or to close the dialog box.

### Errors

Use the CommDlgExtendedError function to retrieve the error value, which may be one of the following:

```
CDERR_FINDRESFAILURE
CDERR_INITIALIZATION
CDERR_LOADRESFAILURE
CDERR_LOADSTRFAILURE
CDERR_LOCKRESFAILURE
CDERR_MEMALLOCFailure
CDERR_MEMLOCKFAILURE
CDERR_NOHINSTANCE
CDERR_NOHOOK
CDERR_NOTEMPLATE
CDERR_STRUCTSIZE
FRERR_BUFFERLENGTHZERO
```

### Comments

The dialog box procedure for the ReplaceText function passes user requests to the application through special messages. The lParam parameter of each of these messages contains a pointer to a FINDREPLACE structure. The procedure sends the messages to the window identified by the hWndOwner member of the FINDREPLACE structure. An application can register the identifier for these messages by specifying the commdlg\_FindReplace string in a call to the RegisterWindowMessage function.

For the TAB key to function correctly, any application that calls the ReplaceText function must also call the IsDialogMessage function in its main message loop. (The IsDialogMessage function returns a value that indicates whether messages are intended for the Replace dialog box.)

### Example

This example initializes a FINDREPLACE structure and calls the ReplaceText function to display the Replace dialog box:

```
FINDREPLACE fr;
char szFindWhat[256] = ""; /* string to find */
char szReplaceWith[256] = ""; /* string to replace */

/* Set all structure fields to zero. */
```

```
memset(&fr, 0, sizeof(FINDREPLACE));

fr.lStructSize = sizeof(FINDREPLACE);
fr.hwndOwner = hwnd;
fr.lpstrFindWhat = szFindWhat;
fr.wFindWhatLen = sizeof(szFindWhat);
fr.lpstrReplaceWith = szReplaceWith;
fr.wReplaceWithLen = sizeof(szReplaceWith);

hDlg = ReplaceText(&fr);
```

In addition to initializing the members of the FINDREPLACE structure and calling the ReplaceText function, an application must register the special FINDMSGSTRING message and process messages from the dialog box. Refer to the description of the FindText function for an example that shows how an application registers and processes a message.

See Also

FindText, IsDialogMessage, RegisterWindowMessage, FINDREPLACE

The following shows how the replace dialog box appears:

### Common dialog box functions (3.1)

<u>CommDlgExtendedError</u>	Retrieves error data for common dialog box procedure
<u>ChooseColor</u>	Creates a color-selection dialog box
<u>ChooseFont function</u>	Creates a font-selection dialog box
<u>FindText</u>	Creates a find-text dialog box
<u>GetFileTitle</u>	Retrieves a filename
<u>GetOpenFileName</u>	Creates an open-filename dialog box
<u>GetSaveFileName</u>	Creates a save-filename dialog box
<u>PrintDlg function</u>	Creates a print-text dialog box
<u>ReplaceText</u>	Creates a replace-text dialog box

## DdeAbandonTransaction (3.1)

```
#include <ddeml.h>

BOOL DdeAbandonTransaction(idInst, hConv, idTransaction)
DWORD idInst;           /* instance identifier */
HCONV hConv;           /* handle of conversation */
/* transaction identifier */
```

The DdeAbandonTransaction function abandons the specified asynchronous transaction and releases all resources associated with the transaction.

Parameter	Description
idInst	Specifies the application-instance identifier obtained by a previous call to the <u>DdeInitialize</u> function.
hConv	Identifies the conversation in which the transaction was initiated. If this parameter is NULL, all transactions are abandoned (the idTransaction parameter is ignored).
idTransaction	Identifies the transaction to terminate. If this parameter is NULL, all active transactions in the specified conversation are abandoned.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_INVALIDPARAMETER
DMLERR_NO_ERROR
DMLERR_UNFOUND_QUEUE_ID
```

### Comments

Only a dynamic data exchange (DDE) client application should call the DdeAbandonTransaction function. If the server application responds to the transaction after the client has called DdeAbandonTransaction, the system discards the transaction results. This function has no effect on synchronous transactions.

### See Also

DdeClientTransaction, DdeGetLastError, DdeInitialize, DdeQueryConvInfo

### DdeAccessData (3.1)

```
#include <ddeml.h>
```

```
BYTE FAR* DdeAccessData(hData, lpcbData)
HDDEDATA hData;          /* handle of global memory object */
DWORD FAR* lpcbData;
/* pointer to variable that receives data length
/
```

The DdeAccessData function provides access to the data in the given global memory object. An application must call the DdeUnaccessData function when it is finished accessing the data in the object.

Parameter	Description
hData	Identifies the global memory object to access.
lpcbData	Points to a variable that receives the size, in bytes, of the global memory object identified by the hData parameter. If this parameter is NULL, no size information is returned.

#### Returns

The return value points to the first byte of data in the global memory object if the function is successful. Otherwise, the return value is NULL.

#### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_INVALIDPARAMETER
DMLERR_NO_ERROR
```

#### Comments

If the hData parameter has not been passed to a Dynamic Data Exchange Management Library (DDEML) function, an application can use the pointer returned by DdeAccessData for read-write access to the global memory object. If hData has already been passed to a DDEML function, the pointer can only be used for read-only access to the memory object.

#### Example

The following example uses the DdeAccessData function to obtain a pointer to a global memory object, uses the pointer to copy data from the object to a local buffer, then frees the pointer:

```
HDDEDATA hData;
LPBYTE lpszAdviseData;
DWORD cbDataLen;
DWORD i;
char szData[128];

lpszAdviseData = DdeAccessData(hData, &cbDataLen);
for (i = 0; i < cbDataLen; i++)
    szData[i] = *lpszAdviseData++;
DdeUnaccessData(hData);
```

#### See Also

DdeAddData, DdeCreateDataHandle, DdeFreeDataHandle, DdeGetLastError, DdeUnaccessData



### DdeAddData (3.1)

```
#include <ddeml.h>
```

```
HDDEDATA DdeAddData(hData, lpvSrcBuf, cbAddData, offObj)
HDDEDATA hData;          /* handle of global memory object */
void FAR* lpvSrcBuf;     /* address of source buffer */
DWORD cbAddData;        /* length of data */
DWORD offObj;           /* offset within global memory object */
```

The DdeAddData function adds data to the given global memory object. An application can add data beginning at any offset from the beginning of the object. If new data overlaps data already in the object, the new data overwrites the old data in the bytes where the overlap occurs. The contents of locations in the object that have not been written to are undefined.

Parameter	Description
hData	Identifies the global memory object that receives additional data.
lpvSrcBuf	Points to a buffer containing the data to add to the global memory object.
cbAddData	Specifies the length, in bytes, of the data to be added to the global memory object.
offObj	Specifies an offset, in bytes, from the beginning of the global memory object. The additional data is copied to the object beginning at this offset.

#### Returns

The return value is a new handle of the global memory object if the function is successful. The new handle should be used in all references to the object. The return value is zero if an error occurs.

#### Errors

Use the [DdeGetLastError](#) function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_INVALIDPARAMETER
DMLERR_MEMORY_ERROR
DMLERR_NO_ERROR
```

#### Comments

After a data handle has been used as a parameter in another Dynamic Data Exchange Management Library (DDEML) function or returned by a DDE callback function, the handle may only be used for read access to the global memory object identified by the handle.

If the amount of global memory originally allocated is not large enough to hold the added data, the DdeAddData function will reallocate a global memory object of the appropriate size.

#### Example

The following example creates a global memory object, uses the DdeAddData function to add data to the object, and then passes the data to a client with an [XTYP\\_POKE](#) transaction:

```
DWORD idInst; /* instance identifier */
HDDEDATA hddeStrings; /* data handle */
HSZ hszMyItem; /* item-name string handle */
DWORD offObj = 0; /* offset in global object */
char szMyBuf[16]; /* temporary string buffer */
HCONV hconv; /* conversation handle */
DWORD dwResult; /* transaction results */
BOOL fAddAString; /* TRUE if strings to add */

/* Create a global memory object. */

hddeStrings = DdeCreateDataHandle(idInst, NULL, 0, 0,
    hszMyItem, CF_TEXT, 0);
```

```

/*
 * If a string is available, the application-defined function
 * IsThereAString() copies it to szMyBuf and returns TRUE. Otherwise,
 * it returns FALSE.
 */

while ((fAddAString = IsThereAString())) {

    /* Add the string to the global memory object. */

    DdeAddData(hddeStrings, /* data handle */
&szMyBuf, /* string buffer */
(DWORD) strlen(szMyBuf) + 1, /* character count */
offObj); /* offset in object */

    offObj = (DWORD) strlen(szMyBuf) + 1; /* adjust offset */
}

/* No more data to add, so poke it to the server. */

DdeClientTransaction((void FAR*) hddeStrings, -1L, hconv, hszMyItem,
CF_TEXT, XTYP_POKE, 1000, &dwResult);

```

See Also

DdeAccessData, DdeCreateDataHandle, DdeGetLastError, DdeUnaccessData

### DdeClientTransaction (3.1)

```
#include <ddeml.h>

HDDDEDATA DdeClientTransaction(lpvData, cbData, hConv, hszItem, uFmt, uType, uTimeout,
                                lpuResult)
void FAR* lpvData;          /* address of data to pass to server */
DWORD cbData;              /* length of data */
HCONV hConv;               /* handle of conversation */
HSZ hszItem;               /* handle of item-name string */
UINT uFmt;                 /* clipboard data format */
UINT uType;                /* transaction type */
DWORD uTimeout;            /* timeout duration */
DWORD FAR* lpuResult;      /* points to transaction result */
```

The DdeClientTransaction function begins a data transaction between a client and a server. Only a dynamic data exchange (DDE) client application can call this function, and only after establishing a conversation with the server.

Parameter	Description								
lpvData	Points to the beginning of the data that the client needs to pass to the server. Optionally, an application can specify the data handle (HDDDEDATA) to pass to the server, in which case the cbData parameter should be set to -1. This parameter is required only if the uType parameter is <u>XTYP_EXECUTE</u> or <u>XTYP_POKE</u> . Otherwise, this parameter should be NULL.								
cbData	Specifies the length, in bytes, of the data pointed to by the lpvData parameter. A value of -1 indicates that lpvData is a data handle that identifies the data being sent.								
hConv	Identifies the conversation in which the transaction is to take place.								
hszItem	Identifies the data item for which data is being exchanged during the transaction. This handle must have been created by a previous call to the <u>DdeCreateStringHandle</u> function. This parameter is ignored (and should be set to NULL) if the uType parameter is <u>XTYP_EXECUTE</u> .								
uFmt	Specifies the standard clipboard format in which the data item is being submitted or requested. For more information about standard clipboard formats, see the <u>Clipboard formats</u> topic.								
uType	Specifies the transaction type. This parameter can be one of the following values: <table border="1" data-bbox="446 1564 1399 1896"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>XTYP_ADVSTART</td> <td>Begins an advise loop. Any number of distinct advise loops can exist within a conversation. An application can alter the advise loop type by combining the <u>XTYP_ADVSTART</u> transaction type with one or more of the following flags: <table border="1" data-bbox="755 1732 1399 1896"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>XTYPF_NODATA</td> <td>Instructs the server to notify the client of any data changes without actually sending the data. This flag gives the client the option of ignoring the</td> </tr> </tbody> </table> </td> </tr> </tbody> </table>	Value	Meaning	XTYP_ADVSTART	Begins an advise loop. Any number of distinct advise loops can exist within a conversation. An application can alter the advise loop type by combining the <u>XTYP_ADVSTART</u> transaction type with one or more of the following flags: <table border="1" data-bbox="755 1732 1399 1896"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>XTYPF_NODATA</td> <td>Instructs the server to notify the client of any data changes without actually sending the data. This flag gives the client the option of ignoring the</td> </tr> </tbody> </table>	Value	Meaning	XTYPF_NODATA	Instructs the server to notify the client of any data changes without actually sending the data. This flag gives the client the option of ignoring the
Value	Meaning								
XTYP_ADVSTART	Begins an advise loop. Any number of distinct advise loops can exist within a conversation. An application can alter the advise loop type by combining the <u>XTYP_ADVSTART</u> transaction type with one or more of the following flags: <table border="1" data-bbox="755 1732 1399 1896"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>XTYPF_NODATA</td> <td>Instructs the server to notify the client of any data changes without actually sending the data. This flag gives the client the option of ignoring the</td> </tr> </tbody> </table>	Value	Meaning	XTYPF_NODATA	Instructs the server to notify the client of any data changes without actually sending the data. This flag gives the client the option of ignoring the				
Value	Meaning								
XTYPF_NODATA	Instructs the server to notify the client of any data changes without actually sending the data. This flag gives the client the option of ignoring the								

		notification or requesting the changed data from the server.
	XTYPF_ACKREQ	Instructs the server to wait until the client acknowledges that it received the previous data item before sending the next data item. This flag prevents a fast server from sending data faster than the client can process it.
	XTYP_ADVSTOP	Ends an advise loop.
	XTYP_EXECUTE	Begins an execute transaction.
	XTYP_POKE	Begins a poke transaction.
	XTYP_REQUEST	Begins a request transaction.
uTimeout		Specifies the maximum length of time, in milliseconds, that the client will wait for a response from the server application in a synchronous transaction. This parameter should be set to TIMEOUT_ASYNC for asynchronous transactions.
lpuResult		Points to a variable that receives the result of the transaction. An application that does not check the result can set this value to NULL. For synchronous transactions, the low-order word of this variable will contain any applicable DDE_ flags resulting from the transaction. This provides support for applications dependent on DDE_APPSTATUS bits. (It is recommended that applications no longer use these bits because they may not be supported in future versions of the DDE Management Library.) For asynchronous transactions, this variable is filled with a unique transaction identifier for use with the <u>DdeAbandonTransaction</u> function and the <u>XTYP_XACT_COMPLETE</u> transaction.

#### Returns

The return value is a data handle that identifies the data for successful synchronous transactions in which the client expects data from the server. The return value is TRUE for successful asynchronous transactions and for synchronous transactions in which the client does not expect data. The return value is FALSE for all unsuccessful transactions.

#### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

DMLERR\_ADVACKTIMEOUT  
DMLERR\_BUSY  
DMLERR\_DATAACKTIMEOUT  
DMLERR\_DLL\_NOT\_INITIALIZED  
DMLERR\_EXECACKTIMEOUT  
DMLERR\_INVALIDPARAMETER  
DMLERR\_MEMORY\_ERROR  
DMLERR\_NO\_CONV\_ESTABLISHED  
DMLERR\_NO\_ERROR  
DMLERR\_NOTPROCESSED  
DMLERR\_POKEACKTIMEOUT  
DMLERR\_POSTMSG\_FAILED  
DMLERR\_REENTRANCY  
DMLERR\_SERVER\_DIED  
DMLERR\_UNADVACKTIMEOUT

#### Comments

When the application is finished using the data handle returned by the DdeClientTransaction function, the application should free the handle by calling the DdeFreeDataHandle function.

Transactions can be synchronous or asynchronous. During a synchronous transaction, the DdeClientTransaction function does not return until the transaction completes successfully or fails. Synchronous transactions cause the client to enter a modal loop while waiting for various asynchronous events. Because of this, the client application can still respond to user input while waiting on a synchronous transaction but cannot begin a second synchronous transaction because of the activity associated with the first. The DdeClientTransaction function fails if any instance of the same task has a synchronous transaction already in progress.

During an asynchronous transaction, the DdeClientTransaction function returns after the transaction is begun, passing a transaction identifier for reference. When the server's DDE callback function finishes processing an asynchronous transaction, the system sends an XTYP\_XACT\_COMPLETE transaction to

the client. This transaction provides the client with the results of the asynchronous transaction that it initiated by calling the `DdeClientTransaction` function. A client application can choose to abandon an asynchronous transaction by calling the `DdeAbandonTransaction` function.

#### Example

The following example requests an advise loop with a DDE server application:

```
HCONV hconv;
HSZ hszNow;
HDDEDATA hData;
DWORD dwResult;

hData = DdeClientTransaction(
    (LPBYTE) NULL, /* pass no data to server */
    0, /* no data */
    hconv, /* conversation handle */
    hszNow, /* item name */
    CF_TEXT, /* clipboard format */
    XTYP_ADVSTART, /* start an advise loop */
    1000, /* time-out in one second */
    &dwResult); /* points to result flags */
```

#### See Also

`DdeAbandonTransaction`, `DdeAccessData`, `DdeConnect`, `DdeConnectList`, `DdeCreateStringHandle`

### DdeCmpStringHandles (3.1)

```
#include <ddeml.h>

int DdeCmpStringHandles(hsz1, hsz2)
HSZ hsz1;      /* handle of first string */
HSZ hsz2;      /* handle of second string */
```

The DdeCmpStringHandles function compares the values of two string handles. The value of a string handle is not related to the case of the associated string.

Parameter	Description
hsz1	Specifies the first string handle.
hsz2	Specifies the second string handle.

#### Returns

The return value can be one of the following:

Value	Meaning
-1	The value of hsz1 is either 0 or less than the value of hsz2.
0	The values of hsz1 and hsz2 are equal (both can be 0).
1	The value of hsz2 is either 0 or less than the value of hsz1.

#### Comments

An application that needs to do a case-sensitive comparison of two string handles should compare the string handles directly. An application should use DdeCompStringHandles for all other comparisons to preserve the case-sensitive nature of dynamic data exchange (DDE).

The DdeCompStringHandles function cannot be used to sort string handles alphabetically.

#### Example

This example compares two service-name string handles and, if the handles are the same, requests a conversation with the server, then issues an X<sub>TYP</sub>\_ADVSTART transaction:

```
HSZ hszClock;    /* service name */
HSZ hszTime;     /* topic name */
HSZ hsz1;        /* unknown server*/
HCONV hConv;     /* conversation handle*/
DWORD dwResult; /* result flags */
DWORD idInst;   /* instance identifier*/

/*
 * Compare unknown service name handle with the string handle
 * for the clock application.
 */

if (!DdeCmpStringHandles(hsz1, hszClock)) {

    /*
    * If this is the clock application, start a conversation
    * with it and request an advise loop.
    */

    hConv = DdeConnect(idInst, hszClock, hszTime, NULL);
    if (hConv != (HCONV) NULL)
        DdeClientTransaction(NULL, 0, hConv, hszNow,
        CF_TEXT, XTYP_ADVSTART, 1000, &dwResult);
}
```

See Also

DdeAccessData, DdeCreateStringHandle, DdeFreeStringHandle

### DdeConnect (3.1)

```
#include <ddeml.h>
```

```
HCONV DdeConnect(idInst, hszService, hszTopic, pCC)
DWORD idInst; /* instance identifier */
HSZ hszService; /* handle of service-name string */
HSZ hszTopic; /* handle of topic-name string */
CONVCONTEXT FAR* pCC; /* address of structure with context data */
```

The DdeConnect function establishes a conversation with a server application that supports the specified service name and topic name pair. If more than one such server exists, the system selects only one.

Parameter	Description
idInst	Specifies the application-instance identifier obtained by a previous call to the <u>DdeInitialize</u> function.
hszService	Identifies the string that specifies the service name of the server application with which a conversation is to be established. This handle must have been created by a previous call to the <u>DdeCreateStringHandle</u> function. If this parameter is NULL, a conversation will be established with any available server.
hszTopic	Identifies the string that specifies the name of the topic on which a conversation is to be established. This handle must have been created by a previous call to the <u>DdeCreateStringHandle</u> function. If this parameter is NULL, a conversation on any topic supported by the selected server will be established.
pCC	Points to the CONVCONTEXT structure that contains conversation-context information. If this parameter is NULL, the server receives the default CONVCONTEXT structure during the <u>XTYP_CONNECT</u> or <u>XTYP_WILDCONNECT</u> transaction.

#### Returns

The return value is the handle of the established conversation if the function is successful. Otherwise, it is NULL.

#### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_INVALIDPARAMETER
DMLERR_NO_CONV_ESTABLISHED
DMLERR_NO_ERROR
```

#### Comments

The client application should not make assumptions regarding which server will be selected. If an instance-specific name is specified in the hszService parameter, a conversation will be established only with the specified instance. Instance-specific service names are passed to an application's dynamic data exchange callback function during the XTYP\_REGISTER and XTYP\_UNREGISTER transactions.

All members of the default CONVCONTEXT structure are set to zero except cb, which specifies the size of the structure, and iCodePage, which specifies CP\_WINANSI (the default code page).

#### Example

The following example creates a service-name string handle and a topic-name string handle, then attempts to establish a conversation with a server that supports the service name and topic name. If the attempt fails, the example retrieves an error value identifying the reason for the failure.

```
DWORD idInst = 0L;
HSZ hszClock;
HSZ hszTime;
HCONV hconv;
UINT uError;
```

```
hszClock = DdeCreateStringHandle(idInst, "Clock", CP_WINANSI);  
hszTime = DdeCreateStringHandle(idInst, "Time", CP_WINANSI);  
  
if ((hconv = DdeConnect(  
    idInst, /* instance identifier */  
    hszClock, /* server's service name */  
    hszTime, /* topic name*/  
    NULL)) == NULL) { /* use default CONVCONTEXT */  
    uError = DdeGetLastError(idInst);  
}
```

See Also

DdeConnectList, DdeCreateStringHandle, DdeDisconnect, DdeDisconnectList, DdeInitialize,  
CONVCONTEXT, XTYP\_CONNECT, XTYP\_REGISTER, XTYP\_UNREGISTER



### DdeConnectList (3.1)

```
#include <ddeml.h>
```

```
HCONVLIST DdeConnectList(idInst, hszService, hszTopic, hConvList, pCC)
DWORD idInst; /* instance identifier */
HSZ hszService; /* handle of service-name string */
HSZ hszTopic; /* handle of topic-name string */
HCONVLIST hConvList; /* handle of conversation list */
CONVCONTEXT FAR* pCC; /* address of structure with context data */
```

The DdeConnectList function establishes a conversation with all server applications that support the specified service/topic name pair. An application can also use this function to enumerate a list of conversation handles by passing the function an existing conversation handle. During enumeration, the Dynamic Data Exchange Management Library (DDEML) removes the handles of any terminated conversations from the conversation list. The resulting conversation list contains the handles of all conversations currently established that support the specified service name and topic name.

Parameter	Description
idInst	Specifies the application-instance identifier obtained by a previous call to the <u>DdeInitialize</u> function.
hszService	Identifies the string that specifies the service name of the server application with which a conversation is to be established. If this parameter is NULL, the system will attempt to establish conversations with all available servers that support the specified topic name.
hszTopic	Identifies the string that specifies the name of the topic on which a conversation is to be established. This handle must have been created by a previous call to the <u>DdeCreateStringHandle</u> function. If this parameter is NULL, the system will attempt to establish conversations on all topics supported by the selected server (or servers).
hConvList	Identifies the conversation list to be enumerated. This parameter should be set to NULL if a new conversation list is to be established.
pCC	Points to the <u>CONVCONTEXT</u> structure that contains conversation-context information. If this parameter is NULL, the server receives the default <u>CONVCONTEXT</u> structure during the <u>XTYP_CONNECT</u> or <u>XTYP_WILDCONNECT</u> transaction.

#### Returns

The return value is the handle of a new conversation list if the function is successful. Otherwise, it is NULL. The handle of the old conversation list is no longer valid.

#### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_INVALID_PARAMETER
DMLERR_NO_CONV_ESTABLISHED
DMLERR_NO_ERROR
DMLERR_SYS_ERROR
```

#### Comments

An application must free the conversation-list handle returned by this function, regardless of whether any conversation handles within the list are active. To free the handle, an application can call the DdeDisconnectList function.

All members of the default CONVCONTEXT structure are set to zero except cb, which specifies the size of the structure, and iCodePage, which specifies CP\_WINANSI (the default code page).

#### Example

The following example uses the `DdeConnectList` function to establish a conversation with all servers that support the System topic, counts the servers, allocates a buffer for storing the server's service-name string handles, and then copies the handles to the buffer:

```
HCONVLIST hconvList; /* conversation list */
DWORD idInst; /* instance identifier*/
HSZ hszSystem; /* System topic */
HCONV hconv = NULL; /* conversation handle*/
CONVINFO ci; /* holds conversation data */
UINT cConv = 0; /* count of conv. handles */
HSZ *pHsz, *aHsz; /* point to string handles */

/* Connect to all servers that support the System topic. */

hconvList = DdeConnectList(idInst, NULL, hszSystem, NULL, NULL);

/* Count the number of handles in the conversation list. */

while ((hconv = DdeQueryNextServer(hconvList, hconv)) != NULL) cConv++;
;

/* Allocate a buffer for the string handles. */

hconv = NULL;
aHsz = (HSZ *) LocalAlloc(LMEM_FIXED, cConv * sizeof(HSZ));

/* Copy the string handles to the buffer. */

pHsz = aHsz;
while ((hconv = DdeQueryNextServer(hconvList, hconv)) != NULL) {
    DdeQueryConvInfo(hconv, QID_SYNC, (PCONVINFO) &ci);
    DdeKeepStringHandle(idInst, ci.hszSvcPartner);
    *pHsz++ = ci.hszSvcPartner;
}

.
. /* Use the handles; converse with servers. */
.

/* Free the memory, and terminate conversations. */

LocalFree((HANDLE) aHsz);
DdeDisconnectList(hconvList);
```

See Also

DdeConnect, DdeCreateStringHandle, DdeDisconnect, DdeDisconnectList, DdeInitialize,  
DdeQueryNextServer, CONVCONTEXT, XTYP\_CONNECT

### DdeCreateDataHandle (3.1)

```
#include <ddeml.h>

HDDDEDATA DdeCreateDataHandle(idInst, lpvSrcBuf, cbInitData, offSrcBuf, hszItem, uFmt, afCmd)
DWORD idInst; /* instance identifier */
void FAR* lpvSrcBuf; /* address of source buffer */
/
DWORD cbInitData; /* length of global memory object */
/
DWORD offSrcBuf; /* offset from beginning of source buffer */
/
HSZ hszItem; /* handle of item-name string */
/
UINT uFmt; /* clipboard data format */
/
UINT afCmd; /* creation flags */
/
```

The DdeCreateDataHandle function creates a global memory object and fills the object with the data pointed to by the lpvSrcBuf parameter. A dynamic data exchange (DDE) application uses this function during transactions that involve passing data to the partner application.

Parameter	Description
idInst	Specifies the application-instance identifier obtained by a previous call to the <u>DdeInitialize</u> function.
lpvSrcBuf	Points to a buffer that contains data to be copied to the global memory object. If this parameter is NULL, no data is copied to the object.
cbInitData	Specifies the amount, in bytes, of memory to allocate for the global memory object. If this parameter is zero, the lpvSrcBuf parameter is ignored.
offSrcBuf	Specifies an offset, in bytes, from the beginning of the buffer pointed to by the lpvSrcBuf parameter. The data beginning at this offset is copied from the buffer to the global memory object.
hszItem	Identifies the string that specifies the data item corresponding to the global memory object. This handle must have been created by a previous call to the <u>DdeCreateStringHandle</u> function. If the data handle is to be used in an <u>XTYP_EXECUTE</u> transaction, this parameter must be set to NULL.
uFmt	Specifies the standard clipboard format of the data.
afCmd	Specifies the creation flags. This parameter can be <u>HDATA_APPOWNED</u> , which specifies that the server application that calls the DdeCreateDataHandle function will own the data handle that this function creates. This makes it possible for the server to share the data handle with multiple clients instead of creating a separate handle for each request. If this flag is set, the server must eventually free the shared memory object associated with this handle by using the <u>DdeFreeDataHandle</u> function. If this flag is not set, after the data handle is returned by the server's <u>DDE</u> callback function or used as a parameter in another DDE Management Library function, the handle becomes invalid in the application that creates the handle.

#### Returns

The return value is a data handle if the function is successful. Otherwise, it is NULL.

#### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_INVALIDPARAMETER
DMLERR_MEMORY_ERROR
DMLERR_NO_ERROR
```

## Comments

Any locations in the global memory object that are not filled are undefined.

After a data handle has been used as a parameter in another DDEML function or has been returned by a DDE callback function, the handle may be used only for read access to the global memory object identified by the handle.

If the application will be adding data to the global memory object (using the DdeAddData function) so that the object exceeds 64K in length, then the application should specify a total length (cbInitData + offSrcData) that is equal to the anticipated maximum length of the object. This avoids unnecessary data copying and memory reallocation by the system.

## Example

The following example processes the XTYP\_WILDCONNECT transaction by returning a data handle to an array of HSZPAIR structures--one for each topic name supported:

```
#define CTOPICS 2

UINT type;
UINT fmt;
HSZPAIR ahp[(CTOPICS + 1)];
HSZ ahszTopicList[CTOPICS];
HSZ hszServ, hszTopic;
WORD i, j;

if (type == XTYP_WILDCONNECT) {

    /*
     * Scan the topic list, and create array of HSZPAIR
     * structures.
     */

        j = 0;
        for (i = 0; i < CTOPICS; i++) {
            if (hszTopic == (HSZ) NULL ||
                hszTopic == ahszTopicList[i]) {
                ahp[j].hszSvc = hszServ;
                ahp[j++].hszTopic = ahszTopicList[i];
            }
        }

        /*
     * End the list with an HSZPAIR structure that contains NULL
     * string handles as its members.
     */

        ahp[j].hszSvc = NULL;
        ahp[j++].hszTopic = NULL;

        /*
     * Return a handle to a global memory object containing the
     * HSZPAIR structures.
     */

        return DdeCreateDataHandle(
            idInst, /* instance identifier */
            &ahp, /* points to HSZPAIR array */
            sizeof(HSZ) * j, /* length of the array */
            0, /* start at the beginning */
            NULL, /* no item-name string */
            fmt, /* return the same format */
            0); /* let the system own it */
}


```

## See Also

DdeAccessData, DdeFreeDataHandle, DdeGetData, DdeInitialize, XTYP\_EXECUTE

## DdeCreateStringHandle (3.1)

```
#include <ddeml.h>
HSZ DdeCreateStringHandle(idInst, lpszString, codepage)
DWORD idInst;          /* instance identifier          */
LPCSTR lpszString;    /* address of null-terminated string */
/*
/
int codepage;
/* code page
/
```

The `DdeCreateStringHandle` function creates a handle that identifies the string pointed to by the `lpszString` parameter. A dynamic data exchange (DDE) client or server application can pass the string handle as a parameter to other DDE Management Library functions.

Parameter	Description
<code>idInst</code>	Specifies the application-instance identifier obtained by a previous call to the <code>DdeInitialize</code> function.
<code>lpszString</code>	Points to a buffer that contains the null-terminated string for which a handle is to be created. This string may be any length.
<code>codepage</code>	Specifies the code page used to render the string. This value should be either <code>CP_WINANSI</code> or the value returned by the <code>GetKBCodePage</code> function. A value of zero implies <code>CP_WINANSI</code> .

### Returns

The return value is a string handle if the function is successful. Otherwise, it is `NULL`.

### Errors

Use the `DdeGetLastError` function to retrieve the error value, which may be one of the following:

```
DMLERR_INVALIDPARAMETER
DMLERR_NO_ERROR
DMLERR_SYS_ERROR
```

### Comments

Two identical strings always correspond to the same string handle. String handles are unique across all tasks that use the DDEML. That is, when an application creates a handle for a string and another application creates a handle for an identical string, the string handles returned to both applications are identical--regardless of case.

The value of a string handle is not related to the case of the string it identifies.

When an application has either created a string handle or received one in the callback function and has used the `DdeKeepStringHandle` function to keep it, the application must free that string handle when it is no longer needed.

An instance-specific string handle is not mappable from string handle to string to string handle again. This is shown in the following example, in which the `DdeQueryString` function creates a string from a string handle and then `DdeCreateStringHandle` creates a string handle from that string, but the two handles are not the same:

```
DWORD idInst;
DWORD cb;
HSZ hszInst, hszNew;
PSZ pszInst;

DdeQueryString(idInst, hszInst, pszInst, cb, CP_WINANSI);
hszNew = DdeCreateStringHandle(idInst, pszInst, CP_WINANSI);
/* hszNew != hszInst ! */
```

### Example

The following example creates a service-name string handle and a topic-name string handle and then attempts to establish a conversation with a server that supports the service name and topic name. If the attempt fails, the example obtains an error value identifying the reason for the failure.

```

DWORD idInst = 0L;
HSZ hszClock;
HSZ hszTime;
HCONV hconv;
UINT uError;

hszClock = DdeCreateStringHandle(idInst, "Clock", CP_WINANSI);
hszTime = DdeCreateStringHandle(idInst, "Time", CP_WINANSI);

if ((hconv = DdeConnect(
    idInst, /* instance identifier */
    hszClock, /* server's service name */
    hszTime, /* topic name*/
    NULL)) == NULL) { /* use default CONVCONTEXT */

    uError = DdeGetLastError(idInst);
}

```

See Also

[DdeAccessData](#), [DdeCmpStringHandles](#), [DdeFreeStringHandle](#), [DdeInitialize](#), [DdeKeepStringHandle](#),  
[DdeQueryString](#)

## DdeDisconnect (3.1)

#include <ddeml.h>

```
BOOL DdeDisconnect(hConv)
HCONV hConv;      /* handle of conversation */
```

The DdeDisconnect function terminates a conversation started by either the DdeConnect or DdeConnectList function and invalidates the given conversation handle.

Parameter	Description
hConv	Identifies the active conversation to be terminated.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_NO_CONV_ESTABLISHED
DMLERR_NO_ERROR
```

### Comments

Any incomplete transactions started before calling DdeDisconnect are immediately abandoned. The XTYP\_DISCONNECT transaction type is sent to the dynamic data exchange (DDE) callback function of the partner in the conversation. Generally, only client applications need to terminate conversations.

### See Also

DdeConnect, DdeConnectList, DdeDisconnectList, XTYP\_DISCONNECT

## DdeDisconnectList (3.1)

#include <ddeml.h>

```
BOOL DdeDisconnectList(hConvList)
HCONVLIST hConvList;          /* handle of conversation list */
```

The DdeDisconnectList function destroys the given conversation list and terminates all conversations associated with the list.

Parameter	Description
hConvList	Identifies the conversation list. This handle must have been created by a previous call to the <u>DdeConnectList</u> function.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_INVALIDPARAMETER
DMLERR_NO_ERROR
```

### Comments

An application can use the DdeDisconnect function to terminate individual conversations in the list.

### See Also

DdeConnect, DdeConnectList, DdeDisconnect, XTYP\_DISCONNECT



### DdeEnableCallback (3.1)

```
#include <ddeml.h>

BOOL DdeEnableCallback(idInst, hConv, uCmd)
DWORD idInst;          /* instance identifier          */
HCONV hConv;           /* handle of conversation      */
                        /*                               */
                        /*                               */
                        /*                               */
                        /*                               */
                        /*                               */
UINT uCmd;
/* the enable/disable function code
/
```

The DdeEnableCallback function enables or disables transactions for a specific conversation or for all conversations that the calling application currently has established.

After disabling transactions for a conversation, the system places the transactions for that conversation in a transaction queue associated with the application. The application should reenable the conversation as soon as possible to avoid losing queued transactions.

Parameter	Description
idInst	Specifies the application-instance identifier obtained by a previous call to the <u>DdeInitialize</u> function.
hConv	Identifies the conversation to enable or disable. If this parameter is NULL, the function affects all conversations.

uCmd Specifies the function code. This parameter can be one of the following values:

Value	Meaning
EC_ENABLEALL	Enables all transactions for the specified conversation.
EC_ENABLEONE	Enables one transaction for the specified conversation.
EC_DISABLE	Disables all blockable transactions for the specified conversation.

A server application can disable the following transactions:

XTYP\_ADVSTART  
XTYP\_ADVSTOP  
XTYP\_EXECUTE  
XTYP\_POKE  
XTYP\_REQUEST

A client application can disable the following transactions:

XTYP\_ADVDATA  
XTYP\_XACT\_COMPLETE

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

DMLERR\_DLL\_NOT\_INITIALIZED  
DMLERR\_NO\_ERROR  
DMLERR\_INVALIDPARAMETER

#### Comments

An application can disable transactions for a specific conversation by returning CBR\_BLOCK from its dynamic data exchange (DDE) callback function. When the conversation is reenabled by using the DdeEnableCallback function, the system generates the same transaction as was in process when the conversation was disabled.

#### See Also

DdeConnect, DdeConnectList, DdeDisconnect, DdeInitialize

## DdeFreeDataHandle (3.1)

```
#include <ddeml.h>
```

```
BOOL DdeFreeDataHandle(hData)  
HDDDEDATA hData; /* handle of global memory object */
```

The DdeFreeDataHandle function frees a global memory object and deletes the data handle associated with the object.

Parameter	Description
hData	Identifies the global memory object to be freed. This handle must have been created by a previous call to the <u>DdeCreateDataHandle</u> function or returned by the <u>DdeClientTransaction</u> function.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

```
DMLERR_INVALIDPARAMETER  
DMLERR_NO_ERROR
```

### Comments

An application must call DdeFreeDataHandle under the following circumstances:

- To free a global memory object that the application allocated by calling the DdeCreateDataHandle function if the object's data handle was never passed by the application to another Dynamic Data Exchange Management Library (DDEML) function
- To free a global memory object that the application allocated by specifying the HDATA\_APPOWNED flag in a call to the DdeCreateDataHandle function
- To free a global memory object whose handle the application received from the DdeClientTransaction function

The system automatically frees an unowned object when its handle is returned by a dynamic data exchange (DDE) callback function or used as a parameter in a DDEML function.

### Example

The following example creates a global memory object containing help information, then frees the object after passing the object's handle to the client application:

```
DWORD idInst;  
HSZ hszItem;  
HDDDEDATA hDataHelp;  
  
char szDdeHelp[] = "DDEML test server help:\r\n\"  
    \"\tThe 'Server' (service) and 'Test' (topic) names may change.\r\n\"  
n\"\  
    \"Items supported under the 'Test' topic are:\r\n\"\  
    \"\tCount:\tThis value increments on each data change.\r\n\"\  
    \"\tRand:\tThis value is changed after each data change. \r\n\"\  
    \"\t\tIn Runaway mode, the above items change after a request.\r\n\"  
n\"\  
    \"\tHuge:\tThis is randomly generated text data >64k that the\r\n\"\  
    \"\t\ttest client can verify. It is recalculated on each\r\n\"\  
    \"\t\trequest. This also verifies huge data poked or executed\r\n\"\  
    \"\t\tfrom the test client.\r\n\"\  
    \"\tHelp:\tThis help information. This data is APPOWNED.\r\n\";  
  
/* Create global memory object containing help information. */  
  
if (!hDataHelp) {  
    hDataHelp = DdeCreateDataHandle(idInst, szDdeHelp,  
    strlen(szDdeHelp) + 1, 0, hszItem, CF_TEXT, HDATA_APPOWNED);  
}
```

```
.  
. /* Pass help information to client application. */  
.   
/* Free the global memory object. */  
  
if (hDataHelp)  
DdeFreeDataHandle(hDataHelp);
```

See Also  
[DdeAccessData](#), [DdeCreateDataHandle](#)

### DdeFreeStringHandle (3.1)

```
#include <ddeml.h>
```

```
BOOL DdeFreeStringHandle(idInst, hsz)  
DWORD idInst;      /* instance identifier */  
HSZ hsz;           /* handle of string */
```

The DdeFreeStringHandle function frees a string handle in the calling application.

Parameter	Description
idInst	Specifies the application-instance identifier obtained by a previous call to the <u>DdeInitialize</u> function.
hsz	Identifies the string handle to be freed. This handle must have been created by a previous call to the <u>DdeCreateStringHandle</u> function.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

An application can free string handles that it creates with the DdeCreateStringHandle function but should not free those that the system passed to the application's dynamic data exchange (DDE) callback function or those returned in the CONVINFO structure by the DdeQueryConvInfo function.

#### Example

The following example frees string handles during the X\_TYP\_DISCONNECT transaction:

```
DWORD idInst = 0L;  
HSZ hszClock;  
HSZ hszTime;  
HSZ hszNow;  
UINT type;  
  
if (type == X_TYP_DISCONNECT) {  
    DdeFreeStringHandle(idInst, hszClock);  
    DdeFreeStringHandle(idInst, hszTime);  
    DdeFreeStringHandle(idInst, hszNow);  
  
    return (HDEDEDATA) NULL;  
}
```

#### See Also

DdeCmpStringHandles, DdeCreateStringHandle, DdeInitialize, DdeKeepStringHandle, DdeQueryString

## DdeGetData (3.1)

```
#include <ddeml.h>
```

```
DWORD DdeGetData(hData, pDest, cbMax, offSrc)
HDDDEDATA hData;          /* handle of global memory object */
void FAR* pDest;          /* address of
destination buffer
/
DWORD cbMax;
* amount of data to copy
/
DWORD offSrc;
* offset to beginning of data
/
```

The DdeGetData function copies data from the given global memory object to the specified local buffer.

Parameter	Description
hData	Identifies the global memory object that contains the data to copy.
pDest	Points to the buffer that receives the data. If this parameter is NULL, the DdeGetData function returns the amount, in bytes, of data that would be copied to the buffer.
cbMax	Specifies the maximum amount, in bytes, of data to copy to the buffer pointed to by the pDest parameter. Typically, this parameter specifies the length of the buffer pointed to by pDest.
offSrc	Specifies an offset within the global memory object. Data is copied from the object beginning at this offset.

### Returns

If the pDest parameter points to a buffer, the return value is the size, in bytes, of the memory object associated with the data handle or the size specified in the cbMax parameter, whichever is lower.

If the pDest parameter is NULL, the return value is the size, in bytes, of the memory object associated with the data handle.

### Errors

Use the [DdeGetLastError](#) function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_INVALID_HDDDEDATA
DMLERR_INVALIDPARAMETER
DMLERR_NO_ERROR
```

### Example

The following example copies data from a global memory object to a local buffer and then fills the TIME structure with data from the buffer:

```
HDDDEDATA hData;
char szBuf[32];

typedef struct {
    int hour;
    int minute;
    int second;
} TIME;

DdeGetData(hData, (LPBYTE) szBuf, 32L, 0L);
sscanf(szBuf, "%d:%d:%d", &nTime.hour, &nTime.minute,
&nTime.second);
```

### See Also

[DdeAccessData](#), [DdeCreateDataHandle](#), [DdeFreeDataHandle](#)

## DdeGetLastError (3.1)

```
#include <ddeml.h>
```

```
UINT DdeGetLastError(idInst)  
DWORD idInst; /* instance identifier */
```

The DdeGetLastError function returns the most recent error value set by the failure of a Dynamic Data Exchange Management Library (DDEML) function and resets the error value to DMLERR\_NO\_ERROR.

Parameter	Description
idInst	Specifies the application-instance identifier obtained by a previous call to the <u>DdeInitialize</u> function.

### Returns

The return value is the last error value. Following are the possible DDEML error values:

Value	Meaning
DMLERR_ADVACKTIMEOUT	A request for a synchronous advise transaction has timed out.
DMLERR_BUSY	The response to the transaction caused the DDE_FBUSY bit to be set.
DMLERR_DATAACKTIMEOUT	A request for a synchronous data transaction has timed out.
DMLERR_DLL_NOT_INITIALIZED	A DDEML function was called without first calling the <u>DdeInitialize</u> function, or an invalid instance identifier was passed to a DDEML function.
DMLERR_DLL_USAGE	An application initialized as APPCLASS_MONITOR has attempted to perform a DDE transaction, or an application initialized as APPCMD_CLIENTONLY has attempted to perform server transactions.
DMLERR_EXECACKTIMEOUT	A request for a synchronous execute transaction has timed out.
DMLERR_INVALIDPARAMETER	A parameter failed to be validated by the DDEML. Some of the possible causes are as follows: <ul style="list-style-type: none"><li>The application used a data handle initialized with a different item-name handle than that required by the transaction.</li><li>The application used a data handle that was initialized with a different clipboard data format than that required by the transaction.</li><li>The application used a client-side conversation handle with a server-side function or vice versa.</li><li>The application used a freed data handle or string handle.</li><li>More than one instance of the application used the same object.</li></ul>
DMLERR_LOW_MEMORY	A <u>DDEML</u> application has created a prolonged race condition (where the server application outruns the client), causing large amounts of memory to be consumed.
DMLERR_MEMORY_ERROR	A memory allocation failed.
DMLERR_NO_CONV_ESTABLISHED	A client's attempt to establish a conversation has failed.
DMLERR_NOTPROCESSED	A transaction failed.
DMLERR_POKEACKTIMEOUT	A request for a synchronous poke transaction has timed out.
DMLERR_POSTMSG_FAILED	An internal call to the <u>PostMessage</u> function has failed.
DMLERR_REENTRANCY	An application instance with a synchronous transaction already in progress attempted to initiate another synchronous transaction, or the <u>DdeEnableCallback</u>

function was called from within a DDEML callback function.

DMLERR\_SERVER\_DIED

A server-side transaction was attempted on a conversation that was terminated by the client, or the server terminated before completing a transaction.

DMLERR\_SYS\_ERROR

An internal error has occurred in the DDEML.

DMLERR\_UNADVACKTIMEOUT

A request to end an advise transaction has timed out.

DMLERR\_UNFOUND\_QUEUE\_ID

An invalid transaction identifier was passed to a DDEML function. Once the application has returned from an XTYP\_XACT\_COMPLETE callback, the transaction identifier for that callback is no longer valid.

#### Example

The following example calls the DdeGetLastError function if the DdeCreateDataHandle function fails:

```
DWORD idInst;
HDDEDATA hddeMyData;
HSZPAIR ahszp[2];
HSZ hszClock, hszTime;

/* Create string handles. */

hszClock = DdeCreateStringHandle(idInst, (LPSTR) "Clock",
    CP_WINANSI);
hszTime = DdeCreateStringHandle(idInst, (LPSTR) "Time",
    CP_WINANSI);

/* Copy handles to an HSZPAIR structure. */

ahszp[0].hszSvc = hszClock;
ahszp[0].hszTopic = hszTime;
ahszp[1].hszSvc = (HSZ) NULL;
ahszp[1].hszTopic = (HSZ) NULL;

/* Create a global memory object. */

hddeMyData = DdeCreateDataHandle(idInst, ahszp,
    sizeof(ahszp), 0, NULL, CF_TEXT, 0);
    if (hddeMyData == NULL)

    /*
     * Pass error value to application-defined error handling
     * function.
     */

    HandleError(DdeGetLastError(idInst));
```

See Also

DdeInitialize

### DdeInitialize (3.1)

```
#include <ddeml.h>
```

```
UINT DdeInitialize(lpIdInst, pfnCallback, afCmd, uRes)
DWORD FAR* lpIdInst;          /* address of instance identifier */
PFNCALLBACK pfnCallback;     /* address of callback function */
/
DWORD afCmd;
/* array of command and filter flags */
/
DWORD uRes;
/* reserved */
/
```

The DdeInitialize function registers an application with the Dynamic Data Exchange Management Library (DDEML). An application must call this function before calling any other DDEML function.

Parameter	Description
lpIdInst	Points to the application-instance identifier. At initialization, this parameter should point to 0L. If the function is successful, this parameter points to the instance identifier for the application. This value should be passed as the idInst parameter in all other DDEML dynamic link library, the application should provide a different callback function for each instance.  If lpIdInst points to a nonzero value, this implies a reinitialization of the DDEML. In this case, lpIdInst must point to a valid application-instance identifier.
pfnCallback	Points to the application-defined DDE callback function. This function processes DDE transactions sent by the system. For more information, see the description of the DdeCallback callback function.
afCmd	Specifies an array of APPCMD_ and CBF_ flags. The APPCMD_ flags provide special instructions to the DdeInitialize function. The CBF_ flags set filters that prevent specific types of transactions from reaching the callback function. Using these flags enhances the performance of a DDE application by eliminating unnecessary calls to the callback function.

This parameter can be a combination of the following flags:

Flag	Meaning
APPCLASS_MONITOR	Makes it possible for the application to monitor DDE activity in the system. This flag is for use by DDE monitoring applications. The application specifies the types of DDE activity to monitor by combining one or more monitor flags with the APPCLASS_MONITOR flag. For details, see the following Comments section.
APPCLASS_STANDARD	Registers the application as a standard (nonmonitoring) DDEML application.
APPCMD_CLIENTONLY	Prevents the application from becoming a server in a DDE conversation. The application can be only a client. This flag reduces resource consumption by the DDEML. It includes the functionality of the CBF_FAIL_ALLSVRXACTIONS flag.
APPCMD_FILTERINITS	Prevents the DDEML from sending XTYP_CONNECT and XTYP_WILDCONNECT transactions to the application until the application



	has created its string handles and registered its service names or has turned off filtering by a subsequent call to the <u>DdeNameService</u> or <u>DdeInitialize</u> function. This flag is always in effect when an application calls <u>DdeInitialize</u> for the first time, regardless of whether the application specifies this flag. On subsequent calls to <u>DdeInitialize</u> , not specifying this flag turns off the application's service-name filters; specifying this flag turns on the application's service-name filters.
CBF_FAIL_ALLSVRXACTIONS	Prevents the callback function from receiving server transactions. The system will return <u>DDE_FNOTPROCESSED</u> to each client that sends a transaction to this application. This flag is equivalent to combining all <u>CBF_FAIL_</u> flags.
CBF_FAIL_ADVISES	Prevents the callback function from receiving <u>XTYP_ADVSTART</u> and <u>XTYP_ADVSTOP</u> transactions. The system will return <u>DDE_FNOTPROCESSED</u> to each client that sends an <u>XTYP_ADVSTART</u> or <u>XTYP_ADVSTOP</u> transaction to the server.
CBF_FAIL_CONNECTIONS	Prevents the callback function from receiving <u>XTYP_CONNECT</u> and <u>XTYP_WILDCONNECT</u> transactions.
CBF_FAIL_EXECUTES	Prevents the callback function from receiving <u>XTYP_EXECUTE</u> transactions. The system will return <u>DDE_FNOTPROCESSED</u> to a client that sends an <u>XTYP_EXECUTE</u> transaction to the server.
CBF_FAIL_POKES	Prevents the callback function from receiving <u>XTYP_POKE</u> transactions. The system will return <u>DDE_FNOTPROCESSED</u> to a client that sends an <u>XTYP_POKE</u> transaction to the server.
CBF_FAIL_REQUESTS	Prevents the callback function from receiving <u>XTYP_REQUEST</u> transactions. The system will return <u>DDE_FNOTPROCESSED</u> to a client that sends an <u>XTYP_REQUEST</u> transaction to the server.
CBF_FAIL_SELFCONNECTIONS	Prevents the callback function from receiving <u>XTYP_CONNECT</u> transactions from the application's own instance. This prevents an application from establishing a <u>DDE</u> conversation with its own instance. An application should use this flag if it needs to communicate with other instances of itself but not with itself.
CBF_SKIP_ALLNOTIFICATIONS	Prevents the callback function from receiving any notifications. This flag is

CBF_SKIP_CONNECT_CONFIRMES	equivalent combining all CBF_SKIP_ flags. Prevents the callback function from receiving <u>XTYP_CONNECT_CONFIRM</u> notifications.
CBF_SKIP_DISCONNECTS	Prevents the callback function from receiving <u>XTYP_DISCONNECT</u> notifications.
CBF_SKIP_REGISTRATIONS	Prevents the callback function from receiving <u>XTYP_REGISTER</u> notifications.
CBF_SKIP_UNREGISTRATIONS	Prevents the callback function from receiving <u>XTYP_UNREGISTER</u> notifications.
uRes	Reserved; must be set to 0L.

#### Returns

The return value is one of the following:

DMLERR\_DLL\_USAGE  
DMLERR\_INVALIDPARAMETER  
DMLERR\_NO\_ERROR  
DMLERR\_SYS\_ERROR

#### Comments

An application that uses multiple instances of the DDEML must not pass DDEML objects between instances.

A DDE monitoring application should not attempt to perform DDE (establish conversations, issue transactions, and so on) within the context of the same application instance.

A synchronous transaction will fail with a DMLERR\_REENTRANCY error if any instance of the same task has a synchronous transaction already in progress.

A DDE monitoring application can combine one or more of the following monitor flags with the APPCLASS\_MONITOR flag to specify the types of DDE activity to monitor:

Flag	Meaning
MF_CALLBACKS	Notifies the callback function whenever a transaction is sent to any <u>DDE</u> callback function in the system.
MF_CONV	Notifies the callback function whenever a conversation is established or terminated.
MF_ERRORS	Notifies the callback function whenever a <u>DDE</u> error occurs.
MF_HSZ_INFO	Notifies the callback function whenever a <u>DDE</u> application creates, frees, or increments the use count of a string handle or whenever a string handle is freed as a result of a call to the <u>DdeUninitialize</u> function.
MF_LINKS	Notifies the callback function whenever an advise loop is started or ended.
MF_POSTMSGS	Notifies the callback function whenever the system or an application posts a <u>DDE</u> message.
MF_SENDMSGS	Notifies the callback function whenever the system or an application sends a <u>DDE</u> message.

#### Example

The following example obtains a procedure-instance address for a DDE callback function, then initializes the application with the DDEML.

```
DWORD idInst = 0L;
FARPROC lpDdeProc;

lpDdeProc = MakeProcInstance((FARPROC) DDECallback, hInst);
if (DdeInitialize((LPDWORD) &idInst, (PFNCALLBACK) lpDdeProc,
    APPCMD_CLIENONLY, 0L))
    return FALSE;
```

See Also

DdeClientTransaction, DdeConnect, DdeCreateDataHandle, DdeEnableCallback, DdeNameService,  
DdePostAdvise, DdeUninitialize

## DdeKeepStringHandle (3.1)

```
#include <ddeml.h>
```

```
BOOL DdeKeepStringHandle(idInst, hsz)
DWORD idInst;      /* instance identifier */
HSZ hsz;           /* handle of string */
```

The DdeKeepStringHandle function increments the usage count (increases it by one) associated with the given handle. This function makes it possible for an application to save a string handle that was passed to the application's dynamic data exchange (DDE) callback function. Otherwise, a string handle passed to the callback function is deleted when the callback function returns.

Parameter	Description
idInst	Specifies the application-instance identifier obtained by a previous call to the <u>DdeInitialize</u> function.
hsz	Identifies the string handle to be saved.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Example

The following example is a portion of a DDE callback function that increases the usage count and saves a local copy of two string handles:

```
HSZ hsz1;
HSZ hsz2;
static HSZ hszServerBase;
static HSZ hszServerInst;
DWORD idInst;

case XTYP_REGISTER:
    /* Keep the handles for later use. */

    DdeKeepStringHandle(idInst, hsz1);
    DdeKeepStringHandle(idInst, hsz2);
    hszServerBase = hsz1;
    hszServerInst = hsz2;
    . /* Finish processing the transaction. */
    .
```

### See Also

DdeCreateStringHandle, DdeFreeStringHandle, DdeInitialize, DdeQueryString

## DdeNameService (3.1)

```
#include <ddeml.h>

HDDDEDATA DdeNameService(idInst, hsz1, hszRes, afCmd)
DWORD idInst;      /* instance identifier */
HSZ hsz1;          /* handle of service-
name string
/
HSZ hszRes;
* reserved
/
UINT afCmd;
* service-name flags
/
```

The DdeNameService function registers or unregisters the service names that a dynamic data exchange (DDE) server supports. This function causes the system to send XTYP\_REGISTER or XTYP\_UNREGISTER transactions to other running DDE Management Library (DDEML) client applications.

A server application should call this function to register each service name that it supports and to unregister names that it previously registered but no longer supports. A server should also call this function to unregister its service names just before terminating.

Parameter	Description										
idInst	Specifies the application-instance identifier obtained by a previous call to the <u>DdeInitialize</u> function.										
hsz1	Identifies the string that specifies the service name that the server is registering or unregistering. An application that is unregistering all of its service names should set this parameter to NULL.										
hszRes	Reserved; should be set to NULL.										
afCmd	Specifies the service-name flags. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>DNS_REGISTER</td><td>Registers the given service name.</td></tr><tr><td>DNS_UNREGISTER</td><td>Unregisters the given service name. If the hsz1 parameter is NULL, all service names registered by the server will be unregistered.</td></tr><tr><td>DNS_FILTERON</td><td>Turns on service-name initiation filtering. This filter prevents a server from receiving <u>XTYP_CONNECT</u> transactions for service names that it has not registered. This is the default setting for this filter. If a server application does not register any service names, the application cannot receive <u>XTYP_WILDCONNECT</u> transactions.</td></tr><tr><td>DNS_FILTEROFF</td><td>Turns off service-name initiation filtering. If this flag is set, the server will receive an <u>XTYP_CONNECT</u> transaction whenever another DDE application calls the <u>DdeConnect</u> function, regardless of the service name.</td></tr></tbody></table>	Value	Meaning	DNS_REGISTER	Registers the given service name.	DNS_UNREGISTER	Unregisters the given service name. If the hsz1 parameter is NULL, all service names registered by the server will be unregistered.	DNS_FILTERON	Turns on service-name initiation filtering. This filter prevents a server from receiving <u>XTYP_CONNECT</u> transactions for service names that it has not registered. This is the default setting for this filter. If a server application does not register any service names, the application cannot receive <u>XTYP_WILDCONNECT</u> transactions.	DNS_FILTEROFF	Turns off service-name initiation filtering. If this flag is set, the server will receive an <u>XTYP_CONNECT</u> transaction whenever another DDE application calls the <u>DdeConnect</u> function, regardless of the service name.
Value	Meaning										
DNS_REGISTER	Registers the given service name.										
DNS_UNREGISTER	Unregisters the given service name. If the hsz1 parameter is NULL, all service names registered by the server will be unregistered.										
DNS_FILTERON	Turns on service-name initiation filtering. This filter prevents a server from receiving <u>XTYP_CONNECT</u> transactions for service names that it has not registered. This is the default setting for this filter. If a server application does not register any service names, the application cannot receive <u>XTYP_WILDCONNECT</u> transactions.										
DNS_FILTEROFF	Turns off service-name initiation filtering. If this flag is set, the server will receive an <u>XTYP_CONNECT</u> transaction whenever another DDE application calls the <u>DdeConnect</u> function, regardless of the service name.										

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_DLL_USAGE
DMLERR_INVALIDPARAMETER
DMLERR_NO_ERROR
```

### Comments

The service name identified by the hsz1 parameter should be a base name (that is, the name should contain no instance-specific information). The system generates an instance-specific name and sends it along with

the base name during the XTYP\_REGISTER and XTYP\_UNREGISTER transactions. The receiving applications can then connect to the specific application instance.

#### Example

The following example initializes an application with the DDEML, creates frequently used string handles, and registers the application's service name:

```
HSZ hszClock;
HSZ hszTime;
HSZ hszNow;
HINSTANCE hinst;
DWORD idInst = 0L;
FARPROC lpDdeProc;

/* Initialize the application for the DDEML. */

lpDdeProc = MakeProcInstance((FARPROC) DdeCallback, hinst);
if (!DdeInitialize((LPDWORD) &idInst, (PFNCALLBACK) lpDdeProc,
    APPCMD_FILTERINITS | CBF_FAIL_EXECUTES, 0L)) {

    /* Create frequently used string handles. */

    hszTime = DdeCreateStringHandle(idInst, "Time", CP_WINANSI);
    hszNow = DdeCreateStringHandle(idInst, "Now", CP_WINANSI);
    hszClock = DdeCreateStringHandle(idInst, "Clock", CP_WINANSI);

    /* Register the service name. */

    DdeNameService(idInst, hszClock, (HSZ) NULL, DNS_REGISTER);

}
```

#### See Also

DdeConnect, DdeConnectList, DdeInitialize, XTYP\_REGISTER, XTYP\_UNREGISTER

### DdePostAdvise (3.1)

```
#include <ddeml.h>

BOOL DdePostAdvise(idInst, hszTopic, hszItem)
DWORD idInst;      /* instance identifier */
HSZ hszTopic;      /* handle of topic-name */
string
/
HSZ hszItem;
/* handle of item-name string */
/
```

The DdePostAdvise function causes the system to send an XTYP\_ADVREQ transaction to the calling (server) application's dynamic data exchange (DDE) callback function for each client that has an advise loop active on the specified topic or item name pair. A server application should call this function whenever the data associated with the topic or item name pair changes.

Parameter	Description
idInst	Specifies the application-instance identifier obtained by a previous call to the <u>DdeInitialize</u> function.
hszTopic	Identifies a string that specifies the topic name. To send notifications for all topics with active advise loops, an application can set this parameter to NULL.
hszItem	Identifies a string that specifies the item name. To send notifications for all items with active advise loops, an application can set this parameter to NULL.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_DLL_USAGE
DMLERR_NO_ERROR
```

#### Comments

A server that has nonenumerable topics or items should set the hszTopic and hszItem parameters to NULL so that the system will generate transactions for all active advise loops. The server's DDE callback function returns NULL for any advise loops that do not need to be updated.

If a server calls DdePostAdvise with a topic/item/format name set that includes the set currently being handled in a XTYP\_ADVREQ callback, a stack overflow may result.

#### Example

The following example calls the DdePostAdvise function whenever the time changes:

```
typedef struct { /* tm */
    int hour;
    int minute;
    int second;
} TIME;

TIME tmTime;
DWORD idInst;
HSZ hszTime;
HSZ hszNow;
TIME tmCurTime;

    . /* Fill tmCurTime with the current time. */
    .

/* Check for any change in second, minute, or hour. */
```

```
if ((tmCurTime.second != tmTime.second) ||  
    (tmCurTime.minute != tmTime.minute) ||  
    (tmCurTime.hour    != tmTime.hour)) {  
  
    /* Send the current time to the clients. */  
  
    DdePostAdvise(idInst, hszTime, hszNow);
```

See Also

DdeInitialize, XTYP\_ADVREQ



### DdeQueryConvInfo (3.1)

```
#include <ddeml.h>
```

```
UINT DdeQueryConvInfo(hConv, idTransaction, lpConvInfo)
HCONV hConv; /* handle of conversation */
DWORD idTransaction;
/* transaction identifier
/
CONVINFO FAR* lpConvInfo;
/* address of structure with conversation data
/
```

The `DdeQueryConvInfo` function retrieves information about a dynamic data exchange (DDE) transaction and about the conversation in which the transaction takes place.

Parameter	Description
<code>hConv</code>	Identifies the conversation.
<code>idTransaction</code>	Specifies the transaction. For asynchronous transactions, this parameter should be a transaction identifier returned by the <code>DdeClientTransaction</code> function. For synchronous transactions, this parameter should be <code>QID_SYNC</code> .
<code>lpConvInfo</code>	Points to the <code>CONVINFO</code> structure that will receive information about the transaction and conversation. The <code>cb</code> member of the <code>CONVINFO</code> structure must specify the length of the buffer allocated for the structure.

#### Returns

The return value is the number of bytes copied into the `CONVINFO` structure, if the function is successful. Otherwise, it is zero.

#### Errors

Use the `DdeGetLastError` function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_NO_CONV_ESTABLISHED
DMLERR_NO_ERROR
DMLERR_UNFOUND_QUEUE_ID
```

#### Example

The following example fills a `CONVINFO` structure with information about a synchronous conversation and then obtains the names of the partner application and topic:

```
DWORD idInst;
HCONV hConv;
CONVINFO ci;
WORD wError;
char szSvcPartner[32];
char szTopic[32];
DWORD cchServ, cchTopic;

if (!DdeQueryConvInfo(hConv, QID_SYNC, &ci))
    wError = DdeGetLastError(idInst);

else {
    cchServ = DdeQueryString(idInst, ci.hszSvcPartner,
        (LPSTR) &szSvcPartner, sizeof(szSvcPartner),
        CP_WINANSI);
    cchTopic = DdeQueryString(idInst, ci.hszTopic,
        (LPSTR) &szTopic, sizeof(szTopic),
        CP_WINANSI);
}
```

#### See Also

[DdeConnect](#), [DdeConnectList](#), [DdeQueryNextServer](#), [CONVINFO](#)

### DdeQueryNextServer (3.1)

```
#include <ddeml.h>

HCONV DdeQueryNextServer(hConvList, hConvPrev)
HCONVLIST hConvList;      /* handle of conversation list      */
HCONV hConvPrev;          /* previous conversation handle */
/
```

The DdeQueryNextServer function obtains the next conversation handle in the given conversation list.

Parameter	Description
hConvList	Identifies the conversation list. This handle must have been created by a previous call to the <u>DdeConnectList</u> function.
hConvPrev	Identifies the conversation handle previously returned by this function. If this parameter is NULL, this function returns the first conversation handle in the list.

#### Returns

The return value is the next conversation handle in the list if the list contains any more conversation handles. Otherwise, it is NULL.

#### Example

The following example uses the DdeQueryNextServer function to count the number of conversation handles in a conversation list and to copy the service-name string handles of the servers to a local buffer:

```
HCONVLIST hconvList; /* conversation list */
DWORD idInst; /* instance identifier*/
HSZ hszSystem; /* System topic */
HCONV hconv = NULL; /* conversation handle*/
CONVINFO ci; /* holds conversation data */
UINT cConv = 0; /* count of conv. handles */
HSZ *pHsz, *aHsz; /* point to string handles */

/* Connect to all servers that support the System topic. */
hconvList = DdeConnectList(idInst, NULL, hszSystem, NULL, NULL);

/* Count the number of handles in the conversation list. */
while ((hconv = DdeQueryNextServer(hconvList, hconv)) != NULL) cConv++;

/* Allocate a buffer for the string handles. */
hconv = NULL;
aHsz = (HSZ *) LocalAlloc(LMEM_FIXED, cConv * sizeof(HSZ));

/* Copy the string handles to the buffer. */
pHsz = aHsz;
while ((hconv = DdeQueryNextServer(hconvList, hconv)) != NULL) {
    DdeQueryConvInfo(hconv, QID_SYNC, (PCONVINFO) &ci);
    DdeKeepStringHandle(idInst, ci.hszSvcPartner);
    *pHsz++ = ci.hszSvcPartner;
}

.
. /* Use the handles; converse with servers. */
.

/* Free the memory, and terminate conversations. */
LocalFree((HANDLE) aHsz);
DdeDisconnectList(hconvList);
```

See Also

DdeConnectList, DdeDisconnectList

## DdeQueryString (3.1)

```
#include <ddeml.h>
```

```
DWORD DdeQueryString(idInst, hsz, lpsz, cchMax, codepage)
```

```
DWORD idInst; /* instance identifier */  
HSZ hsz; /* handle of string */
```

```
LPSTR lpsz;
```

```
* address of destination buffer
```

```
/
```

```
DWORD cchMax;
```

```
* length of buffer
```

```
/
```

```
int codepage;
```

```
* code page
```

```
/
```

The DdeQueryString function copies text associated with a string handle into a buffer.

The string returned in the buffer is always null-terminated. If the string is longer than (cchMax - 1), only the first (cchMax - 1) characters of the string are copied.

If the lpsz parameter is NULL, this function obtains the length, in bytes, of the string associated with the string handle. The length does not include the terminating null character.

Parameter	Description
idInst	Specifies the application-instance identifier obtained by a previous call to the <u>DdeInitialize</u> function.
hsz	Identifies the string to copy. This handle must have been created by a previous call to the <u>DdeCreateStringHandle</u> function.
lpsz	Points to a buffer that receives the string. To obtain the length of the string, this parameter should be set to NULL.
cchMax	Specifies the length, in bytes, of the buffer pointed to by the lpsz parameter. If the string is longer than (cchMax - 1), it will be truncated. If the lpsz parameter is set to NULL, this parameter is ignored.
codepage	Specifies the code page used to render the string. This value should be either CP_WINANSI or the value returned by the <u>GetKBCodePage</u> function.

### Returns

The return value is the length, in bytes, of the returned text (not including the terminating null character) if the lpsz parameter specified a valid pointer. The return value is the length of the text associated with the hsz parameter (not including the terminating null character) if the lpsz parameter specified a NULL pointer. The return value is NULL if an error occurs.

### Example

The following example uses the DdeQueryString function to obtain a service name and topic name that a server has registered:

```
UINT type;
```

```
HSZ hsz1;
```

```
HSZ hsz2;
```

```
char szBaseName[16];
```

```
char szInstName[16];
```

```
if (type == X_TYP_REGISTER) {
```

```
    /* Copy the base service name to a buffer. */
```

```
    DdeQueryString(idInst, hsz1, (LPSTR) &szBaseName,  
    sizeof(szBaseName), CP_WINANSI);
```

```
    /* Copy the instance-specific service name to a buffer. */
```

```
    DdeQueryString(idInst, hsz2, (LPSTR) &szInstName,
```

```
    sizeof(szInstName), CP_WINANSI);  
    return (HDEDEDATA) TRUE;  
}
```

See Also

[DdeCmpStringHandles](#), [DdeCreateStringHandle](#), [DdeFreeStringHandle](#), [DdeInitialize](#)

## DdeReconnect (3.1)

```
#include <ddeml.h>
```

```
HCONV DdeReconnect(hConv)
```

```
HCONV hConv; /* handle of conversation to reestablish */
```

The DdeReconnect function allows a client Dynamic Data Exchange Management Library (DDEML) application to attempt to reestablish a conversation with a service that has terminated a conversation with the client. When the conversation is reestablished, the DDEML attempts to reestablish any preexisting advise loops.

Parameter	Description
hConv	Identifies the conversation to be reestablished. A client must have obtained the conversation handle by a previous call to the <u>DdeConnect</u> function.

### Returns

The return value is the handle of the reestablished conversation if the function is successful. The return value is NULL if the function fails.

### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_INVALIDPARAMETER
DMLERR_NO_CONV_ESTABLISHED
DMLERR_NO_ERROR
```

### Example

The following example shows the context within which an application should call the DdeReconnect function:

```
HDDEDATA EXPENTRY DdeCallback(wType, wFmt, hConv, hsz1,
    hsz2, hData, dwData1, dwData2)
WORD wType; /* transaction type */
WORD wFmt; /* clipboard format */
HCONV hConv; /* handle of the conversation */
HSZ hsz1; /* handle of a string*/
HSZ hsz2; /* handle of a string*/
HDDEDATA hData; /* handle of a global memory object */
DWORD dwData1; /* transaction-specific data */
DWORD dwData2; /* transaction-specific data */
{
    BOOL fAutoReconnect;

    switch (wType) {
        case XTP_DISCONNECT:
            if (fAutoReconnect) {
                DdeReconnect(hConv); /* attempt to reconnect */
            }
            return 0;

        . /* Process other transactions. */
        .
    }
}
```

### See Also

DdeConnect, DdeDisconnect

### DdeSetUserHandle (3.1)

```
#include <ddeml.h>
```

```
BOOL DdeSetUserHandle(hConv,id, hUser)
HCONV hConv;      /* handle of conversation */
DWORD id;         /* transaction identifier */
DWORD hUser;
/* application-defined value
/
```

The DdeSetUserHandle function associates an application-defined 32-bit value with a conversation handle and transaction identifier. This is useful for simplifying the processing of asynchronous transactions. An application can use the DdeQueryConvInfo function to retrieve this value.

Parameter	Description
hConv	Identifies the conversation.
id	Specifies the transaction identifier of an asynchronous transaction. An application should set this parameter to QID_SYNC if no asynchronous transaction is to be associated with the hUser parameter.
hUser	Identifies the value to associate with the conversation handle.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Errors

Use the DdeGetLastError function to retrieve the error value, which may be one of the following:

```
DMLERR_DLL_NOT_INITIALIZED
DMLERR_INVALIDPARAMETER
DMLERR_NO_ERROR
DMLERR_UNFOUND_QUEUE_ID
```

#### See Also

DdeQueryConvInfo

## DdeUnaccessData (3.1)

```
#include <ddeml.h>
```

```
BOOL DdeUnaccessData(hData)
```

```
HDDEDATA hData; /* handle of global memory object */
```

The DdeUnaccessData function frees a global memory object. An application must call this function when it is finished accessing the object.

Parameter	Description
hData	Identifies the global memory object.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Errors

Use the [DdeGetLastError](#) function to retrieve the error value, which may be one of the following:

DMLERR\_DLL\_NOT\_INITIALIZED

DMLERR\_INVALIDPARAMETER

DMLERR\_NO\_ERROR

### Example

The following example obtains a pointer to a global memory object, uses the pointer to copy data from the object to a local buffer, and then uses the DdeUnaccessData function to free the object:

```
HDDEDATA hData;  
LPBYTE lpszAdviseData;  
DWORD cbDataLen;  
DWORD i;  
char szData[128];
```

```
lpszAdviseData = DdeAccessData(hData, &cbDataLen);  
for (i = 0; i < cbDataLen; i++)  
    szData[i] = *lpszAdviseData++;  
DdeUnaccessData(hData);
```

### See Also

[DdeAccessData](#), [DdeAddData](#), [DdeCreateDataHandle](#), [DdeFreeDataHandle](#)



## DdeUninitialize (3.1)

#include <ddeml.h>

```
BOOL DdeUninitialize(idInst)
DWORD idInst;    /* instance identifier */
```

The DdeUninitialize function frees all Dynamic Data Exchange Management Library (DDEML) resources associated with the calling application.

Parameter	Description
idInst	Specifies the application-instance identifier obtained by a previous call to the <u>DdeInitialize</u> function.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The DdeUninitialize function terminates any conversations currently open for the application. If the partner in a conversation fails to terminate its end of the conversation, the system may enter a modal loop while it waits for the conversation to terminate. A timeout period is associated with this loop. If the timeout period expires before the conversation has terminated, a message box appears that gives the user the choice of waiting for another timeout period (Retry), waiting indefinitely (Ignore), or exiting the modal loop (Abort).

An application should wait until its windows are no longer visible and its message loop has terminated before calling this function.

### See Also

DdeDisconnect, DdeDisconnectList, DdeInitialize

### DDE functions (3.1)

<u>DdeAbandonTransaction</u>	Abandons an asynchronous transaction
<u>DdeAccessData</u>	Accesses a DDE global memory object
<u>DdeAddData</u>	Adds data to a DDE global memory object
<u>DdeClientTransaction</u>	Begins a DDE data transaction
<u>DdeCmpStringHandles</u>	Compares two DDE string handles
<u>DdeConnect</u>	Establishes a conversation with a server application
<u>DdeConnectList</u>	Establishes multiple DDE conversations
<u>DdeCreateDataHandle</u>	Creates a DDE data handle
<u>DdeCreateStringHandle</u>	Creates a DDE string handle
<u>DdeDisconnect</u>	Terminates a DDE conversation
<u>DdeDisconnectList</u>	Destroys a DDE conversation list
<u>DdeEnableCallback</u>	Enables or disables one or more DDE conversations
<u>DdeFreeDataHandle</u>	Frees a global memory object
<u>DdeFreeStringHandle</u>	Frees a DDE string handle
<u>DdeGetData</u>	Copies data from a global memory object to a buffer
<u>DdeGetLastError</u>	Returns an error value set by a DDEML function
<u>DdeInitialize</u>	Registers an application with the DDEML
<u>DdeKeepStringHandle</u>	Increments the usage count for a string handle
<u>DdeNameService</u>	Registers or unregisters a service name
<u>DdePostAdvise</u>	Prompts a server to send advise data to a client
<u>DdeQueryConvInfo</u>	Retrieves information about a DDE conversation
<u>DdeQueryNextServer</u>	Obtains the next handle in a DDE conversation list
<u>DdeQueryString</u>	Copies string-handle text into a buffer
<u>DdeReconnect</u>	Reestablishes a DDE conversation
<u>DdeSetUserHandle</u>	Associates a user-defined handle with a transaction
<u>DdeUnaccessData</u>	Frees a DDE global memory object
<u>DdeUninitialize</u>	Frees DDEML resources associated with an application

## XTYP\_ADVDATA (3.1)

```
#include <ddeml.h>
```

```
XTYP_ADVDATA  
hszTopic = hsz1; /* handle of topic-name string */  
hszItem = hsz2; /* handle of item-name string */  
hDataAdvise = hData; /* handle of the advise data */
```

A client's DDE callback function can receive this transaction after the client has established an advise loop with a server. This transaction informs the client that the value of the data item has changed.

Parameter	Description
hszTopic	Value of hsz1. Identifies the topic name.
hszItem	Value of hsz2. Identifies the item name.
hDataAdvise	Value of hData. Identifies the data associated with the topic/item name pair. If the client specified the XTYPF_NODATA flag when it requested the advise loop, this parameter is NULL.

### Returns

A DDE callback function should return DDE\_FACK if it processes this transaction, DDE\_FBUSY if it is too busy to process this transaction, or DDE\_FNOTPROCESSED if it denies this transaction.

### Comments

An application need not free the data handle obtained during this transaction. If the application needs to process the data after the callback function returns, however, it must copy the data associated with the data handle. An application can use the DdeGetData function to copy the data.

### See Also

DdeClientTransaction, DdePostAdvise, XTYP\_ADVREQ, XTYP\_ADVSTART, XTYP\_ADVSTOP

## XTYP\_ADVREQ (3.1)

```
#include <ddeml.h>
```

```
XTYP_ADVREQ  
hszTopic = hsz1; /* handle of topic-name string*/  
hszItem = hsz2; /* handle of item-name string */  
cAdvReq = LOWORD(dwData1); /* count of remaining transactions */
```

The system sends this transaction to a server after the server calls the DdePostAdvise function. This transaction informs the server that an advise transaction is outstanding on the specified topic/item name pair and that data corresponding to the topic/item name pair has changed.

Parameter	Description
hszTopic	Value of hsz1. Identifies the topic name.
hszItem	Value of hsz2. Identifies the item name that has changed.
cAdvReq	Value of the low-order word of dwData1. Specifies the count of XTYP_ADVREQ transactions that remain to be processed on the same topic/item/format name set, within the context of the current call to the <u>DdePostAdvise</u> function. If the current XTYP_ADVREQ transaction is the <u>last one</u> , the count is zero. A server can use this count to determine whether to create an HDATA_APPOWNED data handle for the advise data.  If the DDEML issued the XTYP_ADVREQ transaction because of a late-arriving DDE_FACK transaction flag from a client, the low-order word is set to CADV_LATEACK. The DDE_FACK transaction flag arrives late when a server is sending information faster than a client can process it.

### Returns

The server should call the DdeCreateDataHandle function to create a data handle that identifies the changed data and then should return the handle. If the server is unable to complete the transaction, it should return NULL.

### Comments

A server cannot block this transaction type; the CBR\_BLOCK return value is ignored.

### See Also

DdeCreateDataHandle, DdeInitialize, DdePostAdvise, XTYP\_ADVSTART, XTYP\_ADVSTOP, XTYP\_ADVDATA

### XTYP\_ADVSTART (3.1)

```
#include <ddeml.h>
```

```
XTYP_ADVSTART  
hszTopic = hsz1; /* handle of topic-name string */  
hszItem = hsz2; /* handle of item-name string */
```

A server's DDE callback function receives this transaction when a client specifies XTYP\_ADVSTART for the wType parameter of the DdeClientTransaction function. A client uses this transaction to establish an advise loop with a server.

Parameter	Description
hszTopic	Value of hsz1. Identifies the topic name.
hszItem	Value of hsz2. Identifies the item name.

#### Returns

To allow an advise loop on the specified topic/item name pair, a server's DDE callback function should return a nonzero value. To deny the advise loop, it should return zero. If the callback function returns a nonzero value, any subsequent call by the server to the DdePostAdvise function on the same topic/item name pair will cause the system to send a XTYP\_ADVREQ transaction to the server.

#### Comments

If a client requests an advise loop on a topic/item/format name set for which an advise loop is already established, the DDEML does not create a duplicate advise loop. Instead, the DDEML alters the advise loop flags (XTYPF\_ACKREQ and XTYPF\_NODATA) to match the latest request.

If the server application specified the CBF\_FAIL\_ADVERTISES flag in the DdeInitialize function, this transaction is filtered.

#### See Also

DdeClientTransaction, DdeInitialize, DdePostAdvise, XTYP\_ADVREQ, XTYP\_ADVSTOP

## XTYP\_ADVSTOP (3.1)

```
#include <ddeml.h>
```

```
XTYP_ADVSTOP  
hszTopic = hsz1; /* handle of topic-name string */  
hszItem = hsz2; /* handle of item-name string */
```

A server's DDE callback function receives this transaction when a client specifies XTYP\_ADVSTOP for the wType parameter of the DdeClientTransaction function. A client uses this transaction to end an advise loop with a server.

Parameter	Description
hszTopic	Value of hsz1. Identifies the topic name.
hszItem	Value of hsz2. Identifies the item name.

### Returns

This transaction does not return a value.

### Comments

If the server application specified the CBF\_FAIL\_ADVISES flag in the DdeInitialize function, this transaction is filtered.

### See Also

DdeClientTransaction, DdeInitialize, DdePostAdvise, XTYP\_ADVREQ, XTYP\_ADVSTART

## XTYP\_CONNECT (3.1)

```
#include <ddeml.h>
```

```
XTYP_CONNECT  
hszTopic = hsz1; /* handle of topic-name string */  
hszService = hsz2; /* handle of service-name string */  
pcc = (CONVCONTEXT FAR *)dwData1; /* address of CONVCONTEXT structure */  
/  
fSameInst = (BOOL) dwData2; /* same instance flag*/
```

A server's DDE callback function receives this transaction when a client specifies a service name that the server supports and a topic name that is not set to NULL in a call to the DdeConnect function.

Parameter	Description
hszTopic	Value of hsz1. Identifies the topic name.
hszService	Value of hsz2. Identifies the service name.
pcc	Value of dwData1. Points to a <u>CONVCONTEXT</u> data structure that contains context information for the conversation. If the client is not a <u>DDEML</u> application, this parameter should be set to zero.
fSameInst	Value of dwData2. Specifies whether the client is the same application instance as the server. If this parameter is TRUE, the client is the same instance; if this parameter is FALSE, the client is a different instance.

### Returns

To allow the client to establish a conversation on the specified service/topic name pair, a server's DDE callback function should return a nonzero value. To deny the conversation, it should return zero. If the callback function returns a nonzero value and a conversation is successfully established, the system passes the conversation handle to the server by issuing an XTYP\_CONNECT\_CONFIRM transaction to the server's DDE callback function (unless the server specified the CBF\_FAIL\_CONNECT\_CONFIRMS flag in the DdeInitialize function).

### Comments

If the server application specified the CBF\_FAIL\_CONNECTIONS flag in the DdeInitialize function, this transaction is filtered.

A server cannot block this transaction type; the CBR\_BLOCK return value is ignored.

### See Also

DdeConnect, DdeInitialize, CONVCONTEXT, XTYP\_CONNECT\_CONFIRM

## XTYP\_CONNECT\_CONFIRM (3.1)

```
#include <ddeml.h>
```

```
XTYP_CONNECT_CONFIRM  
hszTopic = hsz1; /* handle of topic-name string */  
hszService = hsz2; /* handle of service-name string */  
fSameInst = (BOOL) dwData2; /* same instance flag */
```

A server's DDE callback function receives this transaction to confirm that a conversation has been established with a client and to provide the server with the conversation handle. The system sends this transaction as a result of a previous XTYP\_CONNECT or XTYP\_WILDCONNECT transaction.

Parameter	Description
hszTopic	Value of hsz1. Identifies the topic name on which the conversation has been established.
hszService	Value of hsz2. Identifies the service name on which the conversation has been established.
fSameInst	Value of dwData2. Specifies whether the client is the same application instance as the server. If this parameter is a nonzero value, the client is the same instance. If this parameter is zero, the client is a different instance.

### Returns

This transaction does not return a value.

### Comments

If the server application specified the CBF\_FAIL\_CONFIRMS flag in the DdeInitialize function, this transaction is filtered.

A server cannot block this transaction type; the CBR\_BLOCK return value is ignored.

### See Also

DdeConnect, DdeConnectList, DdeInitialize, XTYP\_CONNECT, XTYP\_WILDCONNECT



## XTYP\_DISCONNECT (3.1)

```
#include <ddeml.h>
```

```
XTYP_DISCONNECT  
fSameInst = (BOOL) dwData2; /* same instance flag */
```

An application's DDE callback function receives this transaction when the application's partner in a conversation uses the DdeDisconnect function to terminate the conversation.

Parameter	Description
fSameInst	Value of dwData2. Specifies whether the partners in the conversation are the same application instance. If this parameter is TRUE, the partners are the same instance. If this parameter is FALSE, the partners are different instances.

### Returns

This transaction does not return a value.

### Comments

If the application specified the CBF\_SKIP\_DISCONNECTS flag in the DdeInitialize function, this transaction is filtered.

The application can obtain the status of the terminated conversation by calling the DdeQueryConvInfo function while processing this transaction. The conversation handle becomes invalid after the callback function returns.

An application cannot block this transaction type; the CBR\_BLOCK return value is ignored.

### See Also

DdeDisconnect, DdeQueryConvInfo

## XTYP\_ERROR (3.1)

```
#include <ddeml.h>
```

```
XTYP_ERROR  
wErr = LOWORD(dwData1); /* error value */
```

A DDE callback function receives this transaction when a critical error occurs.

Parameter	Description
wErr	Value of dwData1. Specifies the error value. Currently, only the DMLERR_LOW_MEMORY error value is supported. It means that memory is low--advise, poke, or execute data may be lost, or the system may fail.

### Returns

This transaction does not return a value.

### Comments

An application cannot block this transaction type; the CBR\_BLOCK return value is ignored. The DDEML attempts to free memory by removing noncritical resources. An application that has blocked conversations should unblock them.

## XTYP\_EXECUTE (3.1)

```
#include <ddeml.h>
```

```
XTYP_EXECUTE
```

```
hszTopic = hsz1;    /* handle of the topic-name string */  
hDataCmd = hData;  /* handle of the command string   */
```

A server's DDE callback function receives this transaction when a client specifies XTYP\_EXECUTE for the wType parameter of the DdeClientTransaction function. A client uses this transaction to send a command string to the server.

Parameter	Description
hszTopic	Value of hsz1. Identifies the topic name.
hDataCmd	Value of hData. Identifies the command string.

### Returns

A server's DDE callback function should return DDE\_FAIL if it processes this transaction, DDE\_FBUSY if it is too busy to process this transaction, or DDE\_FNOTPROCESSED if it denies this transaction.

### Comments

If the server application specified the CBF\_FAIL\_EXECUTES flag in the DdeInitialize function, this transaction is filtered.

An application need not free the data handle obtained during this transaction. If the application needs to process the string after the callback function returns, however, the application must copy the command string associated with the data handle. An application can use the DdeGetData function to copy the data.

### See Also

DdeClientTransaction, DdeInitialize

## XTYP\_MONITOR (3.1)

```
#include <ddeml.h>
```

```
XTYP_MONITOR  
hDataEvent = hData; /* handle of event data */  
fwEvent = dwData2; /* event flag */
```

The DDE callback function of a DDE debugging application receives this transaction whenever a DDE event occurs in the system. An application can receive this transaction only if it specified the APPCLASS\_MONITOR flag when it called the DdeInitialize function.

Parameter	Description																
hDataEvent	Value of hData. Identifies a global memory object that contains information about the <u>DDE</u> event. The application should use the <u>DdeAccessData</u> function to obtain a pointer to the object.																
fwEvent	Value of dwData2. Specifies the <u>DDE</u> event. This parameter may be one of the following values:																
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MF_CALLBACKS</td><td>The system sent a transaction to a <u>DDE</u> callback function. The global memory object contains a <u>MONCBSTRUCT</u> structure that provides information about the transaction.</td></tr><tr><td>MF_CONV</td><td>A <u>DDE</u> conversation was established or terminated. The global memory object contains a <u>MONCONVSTRUCT</u> structure that provides information about the conversation.</td></tr><tr><td>MF_ERRORS</td><td>A <u>DDE</u> error occurred. The global memory object contains a <u>MONERRSTRUCT</u> structure that provides information about the error.</td></tr><tr><td>MF_HSZ_INFO</td><td>A <u>DDE</u> application created or freed a string handle or incremented the use count of a string handle, or a string handle was freed as a result of a call to the <u>DdeUninitialize</u> function. The global memory object contains a <u>MONHSZSTRUCT</u> structure that provides information about the string handle.</td></tr><tr><td>MF_LINKS</td><td>A <u>DDE</u> application started or ended an advise loop. The global memory object contains a <u>MONLINKSTRUCT</u> structure that provides information about the advise loop.</td></tr><tr><td>MF_POSTMSGS</td><td>The system or an application posted a <u>DDE</u> message. The global memory object contains a <u>MONMSGSTRUCT</u> structure that provides information about the message.</td></tr><tr><td>MF_SENDMSGS</td><td>The system or an application sent a <u>DDE</u> message. The global memory object contains a <u>MONMSGSTRUCT</u> structure that provides information about the message.</td></tr></tbody></table>	Value	Meaning	MF_CALLBACKS	The system sent a transaction to a <u>DDE</u> callback function. The global memory object contains a <u>MONCBSTRUCT</u> structure that provides information about the transaction.	MF_CONV	A <u>DDE</u> conversation was established or terminated. The global memory object contains a <u>MONCONVSTRUCT</u> structure that provides information about the conversation.	MF_ERRORS	A <u>DDE</u> error occurred. The global memory object contains a <u>MONERRSTRUCT</u> structure that provides information about the error.	MF_HSZ_INFO	A <u>DDE</u> application created or freed a string handle or incremented the use count of a string handle, or a string handle was freed as a result of a call to the <u>DdeUninitialize</u> function. The global memory object contains a <u>MONHSZSTRUCT</u> structure that provides information about the string handle.	MF_LINKS	A <u>DDE</u> application started or ended an advise loop. The global memory object contains a <u>MONLINKSTRUCT</u> structure that provides information about the advise loop.	MF_POSTMSGS	The system or an application posted a <u>DDE</u> message. The global memory object contains a <u>MONMSGSTRUCT</u> structure that provides information about the message.	MF_SENDMSGS	The system or an application sent a <u>DDE</u> message. The global memory object contains a <u>MONMSGSTRUCT</u> structure that provides information about the message.
Value	Meaning																
MF_CALLBACKS	The system sent a transaction to a <u>DDE</u> callback function. The global memory object contains a <u>MONCBSTRUCT</u> structure that provides information about the transaction.																
MF_CONV	A <u>DDE</u> conversation was established or terminated. The global memory object contains a <u>MONCONVSTRUCT</u> structure that provides information about the conversation.																
MF_ERRORS	A <u>DDE</u> error occurred. The global memory object contains a <u>MONERRSTRUCT</u> structure that provides information about the error.																
MF_HSZ_INFO	A <u>DDE</u> application created or freed a string handle or incremented the use count of a string handle, or a string handle was freed as a result of a call to the <u>DdeUninitialize</u> function. The global memory object contains a <u>MONHSZSTRUCT</u> structure that provides information about the string handle.																
MF_LINKS	A <u>DDE</u> application started or ended an advise loop. The global memory object contains a <u>MONLINKSTRUCT</u> structure that provides information about the advise loop.																
MF_POSTMSGS	The system or an application posted a <u>DDE</u> message. The global memory object contains a <u>MONMSGSTRUCT</u> structure that provides information about the message.																
MF_SENDMSGS	The system or an application sent a <u>DDE</u> message. The global memory object contains a <u>MONMSGSTRUCT</u> structure that provides information about the message.																

### Returns

The callback function should return zero if it processes this transaction.

### See Also

DdeAccessData, DdeInitialize, MONCBSTRUCT, MONCONVSTRUCT, MONERRSTRUCT, MONHSZSTRUCT, MONLINKSTRUCT, MONMSGSTRUCT

## XTYP\_POKE (3.1)

```
#include <ddeml.h>
```

```
XTYP_POKE  
hszTopic = hsz1; /* handle of topic-name string */  
hszItem = hsz2; /* handle of item-name string */  
hDataPoke = hData; /* handle of data for server */
```

A server's DDE callback function receives this transaction when a client specifies XTYP\_POKE as the wType parameter of the DdeClientTransaction function. A client uses this transaction to send unsolicited data to the server.

Parameter	Description
hszTopic	Value of hsz1. Identifies the topic name.
hszItem	Value of hsz2. Identifies the item name.
hDataPoke	Value of hData. Identifies the data that the client is sending to the server.

### Returns

A server's DDE callback function should return DDE\_FACK if it processes this transaction, DDE\_FBUSY if it is too busy to process this transaction, or DDE\_FNOTPROCESSED if it denies this transaction.

### Comments

If the server application specified the CBF\_FAIL\_POKES flag in the DdeInitialize function, this transaction is filtered.

### See Also

DdeClientTransaction, DdeInitialize

## XYP\_REGISTER (3.1)

```
#include <ddeml.h>
```

```
XYP_REGISTER
```

```
hszBaseServName = hsz1; /* handle of base service-name string*/  
hszInstServName = hsz2; /* handle of instance service-name string */
```

A DDE callback function receives this transaction type whenever a DDEML server application uses the DdeNameService function to register a service name or whenever a non-DDEML application that supports the System topic is started.

Parameter	Description
hszBaseServName	Value of hsz1. Identifies the base service name being registered.
hszInstServName	Value of hsz2. Identifies the instance-specific service name being registered.

Returns

This transaction does not return a value.

Comments

If the application specified the CBF\_SKIP\_REGISTRATIONS flag in the DdeInitialize function, this transaction is filtered.

An application cannot block this transaction type; the CBR\_BLOCK return value is ignored.

An application should use the hszBaseServName parameter to add the service name to the list of servers available to the user. An application should use the hszInstServName parameter to identify which application instance has started.

See Also

DdeInitialize, DdeNameService, XYP\_UNREGISTER

## XTYP\_REQUEST (3.1)

```
#include <ddeml.h>
```

```
XTYP_REQUEST
```

```
hszTopic = hsz1; /* handle of topic-name string */
```

```
hszItem = hsz2; /* handle of item-name string */
```

A DDE server callback function receives this transaction when a client specifies XTYP\_REQUEST for the wType parameter of the DdeClientTransaction function. A client uses this transaction to request data from a server.

Parameter	Description
hszTopic	Value of hsz1. Identifies the topic name.
hszItem	Value of hsz2. Identifies the item name that has changed.

### Returns

The server should call the DdeCreateDataHandle function to create a data handle that identifies the changed data and then should return the handle. The server should return NULL if it is unable to complete the transaction. If the server returns NULL, the client receives a DDE\_FNOTPROCESSED acknowledgment flag.

### Comments

If the server application specified the CBF\_FAIL\_REQUESTS flag in the DdeInitialize function, this transaction is filtered.

If responding to this transaction requires lengthy processing, the server can return CBR\_BLOCK to suspend future transactions on the current conversation and then process the transaction asynchronously. When the server has finished and the data is ready to pass to the client, the server can call the DdeEnableCallback function to resume the conversation.

### See Also

DdeClientTransaction, DdeCreateDataHandle, DdeEnableCallback, DdeInitialize

## XTYP\_UNREGISTER (3.1)

```
#include <ddeml.h>
```

```
XTYP_UNREGISTER
```

```
hszBaseServName = hsz1; /* handle of base service-name string*/  
hszInstServName = hsz2; /* handle of instance service-name string */
```

A DDE callback function receives this transaction type whenever a DDEML server application uses the DdeNameService function to unregister a service name or whenever a non-DDEML application that supports the System topic is terminated.

Parameter	Description
hszBaseServName	Value of hsz1. Identifies the base service name being unregistered.
hszInstServName	Value of hsz2. Identifies the instance-specific service name being unregistered.

Returns

This transaction does not return a value.

Comments

If the application specified the CBF\_SKIP\_REGISTRATIONS flag in the DdeInitialize function, this transaction is filtered.

An application cannot block this transaction type; the CBR\_BLOCK return value is ignored.

An application should use the hszBaseServName parameter to remove the service name from the list of servers available to the user. An application should use the hszInstServName parameter to identify which application instance has terminated.

See Also

DdeInitialize, DdeNameService, XTYP\_REGISTER



## XTYP\_WILDCONNECT (3.1)

```
#include <ddeml.h>
```

```
XTYP_WILDCONNECT  
hszTopic = hsz1; /* handle of topic-name string */  
hszService = hsz2; /* handle of service-name string */  
pcc = (CONVCONTEXT FAR *)dwData1; /* address of CONVCONTEXT structure */  
/  
fSameInst = (BOOL) dwData2; /* same-instance flag*/
```

A server's DDE callback function receives this transaction when a client specifies a service name that is set to NULL, a topic name that is set to NULL, or both in a call to the DdeConnect function. This transaction allows a client to establish a conversation on each of the server's service/topic name pairs that matches the specified service name and topic name.

Parameter	Description
hszTopic	Value of hsz1. Identifies the topic name. If this parameter is <u>NULL</u> , the client is requesting a conversation on all topic names that the server supports.
hszService	Value of hsz2. Identifies the service name. If this parameter is <u>NULL</u> , the client is requesting a conversation on all service names that the server supports.
pcc	Value of dwData1. Points to a <u>CONVCONTEXT</u> data structure that contains context information for the conversation. If the client is not a <u>DDEML</u> application, this parameter is set to zero.
fSameInst	Value of dwData2. Specifies whether the client is the same application instance as the server. If this parameter is <u>TRUE</u> , the client is same instance. If this parameter is <u>FALSE</u> , the client is a different instance.

### Returns

The server should return a data handle that identifies an array of HSZPAIR structures. The array should contain one structure for each service/topic name pair that matches the service/topic name pair requested by the client. The array must be terminated by a NULL string handle. The system sends the XTYP\_CONNECT\_CONFIRM transaction to the server to confirm each conversation and to pass the conversation handles to the server. If the server specified the CBF\_SKIP\_CONNECT\_CONFIRMS flag in the DdeInitialize function, it cannot receive these confirmations.

To refuse the XTYP\_WILDCONNECT transaction, the server should return NULL.

### Comments

If the server application specified the CBF\_FAIL\_CONNECTIONS flag in the DdeInitialize function, this transaction is filtered.

A server cannot block this transaction type; the CBR\_BLOCK return code is ignored.

### See Also

DdeConnect, DdeInitialize, CONVCONTEXT, XTYP\_CONNECT\_CONFIRM

## XTYP\_XACT\_COMPLETE (3.1)

```
#include <ddeml.h>
```

```
XTYP_XACT_COMPLETE  
hszTopic = hsz1; /* handle of topic-name string */  
hszItem = hsz2; /* handle of item-name string */  
hDataXact = hData; /* handle of transaction data */  
dwXactID = dwData1; /* transaction identifier */  
fwStatus = dwData2; /* status flag */
```

A DDE client callback function receives this transaction when an asynchronous transaction, initiated by a call to the DdeClientTransaction function, has concluded.

Parameter	Description
hszTopic	Value of hsz1. Identifies the topic name involved in the completed transaction.
hszItem	Value of hsz2. Identifies the item name involved in the completed transaction.
hDataXact	Value of hData. Identifies the data involved in the completed transaction, if applicable. If the transaction was successful but involved no data, this parameter is TRUE. If the transaction was unsuccessful, this parameter is NULL.
dwXactID	Value of dwData1. Contains the transaction identifier of the completed transaction.
fwStatus	Value of dwData2. Contains any applicable DDE_ status flags in the low-order word. This provides support for applications dependent on DDE_APPSTATUS bits. It is recommended that applications no longer use these bits--future versions of the <u>DDEML</u> may not support them.

### Returns

This transaction does not return a value.

### Comments

An application need not free the data handle obtained during this transaction. If the application needs to process the data after the callback function returns, however, the application must copy the data associated with the data handle. An application can use the DdeGetData function to copy the data.

### See Also

DdeClientTransaction

DDE transactions

<u>XTYP_ADVDATA</u>	Passes advise data to a client
<u>XTYP_ADVREQ</u>	Prompts a server to send advise data to a client
<u>XTYP_ADVSTART</u>	Establishes an advise loop with a server
<u>XTYP_ADVSTOP</u>	Ends an advise loop with a server
<u>XTYP_CONNECT</u>	Requests a DDE conversation with a client
<u>XTYP_CONNECT_CONFIRM</u>	Confirms a DDE conversation with a client
<u>XTYP_DISCONNECT</u>	Terminates a DDE conversation
<u>XTYP_ERROR</u>	Notifies a DDEML application of a critical error
<u>XTYP_EXECUTE</u>	Executes a server command
<u>XTYP_MONITOR</u>	Informs a DDE debugging application of a DDE event
<u>XTYP_POKE</u>	Sends unsolicited data to a server
<u>XTYP_REGISTER</u>	Registers a service name
<u>XTYP_REQUEST</u>	Requests data from a server
<u>XTYP_UNREGISTER</u>	Unregisters a service name
<u>XTYP_WILDCONNECT</u>	Requests multiple DDE conversations
<u>XTYP_XACT_COMPLETE</u>	Confirms completion of an asynchronous transaction

## DragAcceptFiles (3.1)

```
#include shellapi.h

void DragAcceptFiles(HWND hwnd, BOOL fAccept)
/* handle of the registering window */
/* flag
for whether dropped files are accepted
/
```

The DragAcceptFiles function registers whether a given window accepts dropped files.

Parameter	Description
hwnd	Identifies the window registering whether it accepts dropped files.
fAccept	Specifies whether the window specified by the hwnd parameter accepts dropped files. An application should set this value to TRUE to accept dropped files or FALSE to discontinue accepting dropped files.

### Returns

This function does not return a value.

### Comments

When an application calls DragAcceptFiles with fAccept set to TRUE, Windows File Manager (WINFILE.EXE) sends the specified window a WM\_DROPFILES message each time the user drops a file in that window.

### See Also

WM\_DROPFILES

### DragFinish (3.1)

```
#include shellapi.h
```

```
void DragFinish(hDrop)
```

```
HDROP hDrop; /* handle of memory to free */
```

The DragFinish function releases memory that Windows allocated for use in transferring filenames to the application.

Parameter	Description
hDrop	Identifies the internal data structure that describes dropped files. This handle is passed to the application in the wParam parameter of the <u>WM_DROPFILES</u> message.

#### Returns

This function does not return a value.

#### See Also

WM\_DROPFILES

### DragQueryFile (3.1)

```
#include shellapi.h
```

```
UINT DragQueryFile(hDrop, iFile, lpszFile, cb)
```

```
HDROP hDrop; /* handle of structure for dropped files */
```

```
UINT iFile; /* index of file
```

```
to query
```

```
/
```

```
LPSTR lpszFile;
```

```
* address of buffer for returned filename
```

```
/
```

```
UINT cb;
```

```
* size of buffer for filename
```

```
/
```

The DragQueryFile function retrieves the number of dropped files and their filenames.

Parameter	Description
hDrop	Identifies the internal data structure containing filenames for the dropped files. This handle is passed to the application in the wParam parameter of the <u>WM_DROPFILES</u> message.
iFile	Specifies the index of the file to query. The index of the first file is 0. If the value of the iFile parameter is -1, DragQueryFile returns the number of files dropped. If the value of the iFile parameter is between zero and the total number of files dropped, DragQueryFile copies the filename corresponding to that value to the buffer pointed to by the lpszFile parameter.
lpszFile	Points to a null-terminated string that contains the filename of a dropped file when the function returns. If this parameter is NULL and the iFile parameter specifies the index for the name of a dropped file, DragQueryFile returns the required size, in bytes, of the buffer for that filename.
cb	Specifies the size, in bytes, of the lpszFile buffer.

#### Returns

When the function copies a filename to the lpszFile buffer, the return value is the number of bytes copied. If the iFile parameter is 0xFFFF, the return value is the number of dropped files. If iFile is between zero and the total number of dropped files and if lpszFile is NULL, the return value is the required size of the lpszFile buffer.

#### See Also

DragQueryPoint, WM\_DROPFILES

## DragQueryPoint (3.1)

#include shellapi.h

```
BOOL DragQueryPoint(hDrop, lppt)
```

```
HDROP hDrop;          /* handle of structure for dropped file */
```

```
POINT FAR* lppt;      /
```

```
/* address of structure for cursor coordinates
```

```
/
```

The DragQueryPoint function retrieves the window coordinates of the cursor when a file is dropped.

Parameter	Description
hDrop	Identifies the internal data structure that describes the dropped file. This structure is returned in the wParam parameter of the <u>WM_DROPFILES</u> message.
lppt	Points to a <u>POINT</u> structure that the function fills with the coordinates of the position at which the cursor was located when the file was dropped.

### Returns

The return value is nonzero if the file is dropped in the client area of the window. Otherwise, it is zero.

### Comments

The DragQueryPoint function fills the POINT structure with the coordinates of the position at which the cursor was located when the user released the left mouse button. The window for which coordinates are returned is the window that received the WM\_DROPFILES message.

### See Also

DragQueryFile, POINT, WM\_DROPFILES

### Drag-drop functions (3.1)

<u>DragAcceptFiles</u>	Registers whether a window accepts dropped files
<u>DragFinish</u>	Releases memory allocated for dropping files
<u>DragQueryFile</u>	Retrieves the filename of a dropped file
<u>DragQueryPoint</u>	Retrieves the mouse position when a file is dropped



### FMExtensionProc (3.1)

```
#include <wfext.h>
HMENU FAR PASCAL FMExtensionProc(hwnd, wParam, lParam)
HWND hwnd;          /* handle of the extension window */
WORD wParam;        /* menu-item identifier or
message */
LONG lParam;        /* additional message information */
```

The FMExtensionProc function, an application-defined callback function, processes menu commands and messages sent to a File Manager extension dynamic-link library (DLL).

Parameter	Description														
hwnd	Identifies the File Manager window. An extension DLL should use this handle to specify the parent for any dialog boxes or message boxes that the DLL may display and to send request messages to File Manager.														
wParam	Specifies the message. This parameter may be one of the following values:														
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>1-99</td><td>Identifier for the menu item that the user selected.</td></tr><tr><td>FMEVENT_INITMENU</td><td>User selected the extension's menu.</td></tr><tr><td>FMEVENT_LOAD</td><td>File Manager is loading the extension DLL.</td></tr><tr><td>FMEVENT_SELCHANGE</td><td>Selection in File Manager's directory window, or Search Results window, changed.</td></tr><tr><td>FMEVENT_UNLOAD</td><td>File Manager is unloading the extension DLL.</td></tr><tr><td>FMEVENT_USER_REFRESH</td><td>User chose the Refresh command from the Window menu.</td></tr></tbody></table>	Value	Meaning	1-99	Identifier for the menu item that the user selected.	FMEVENT_INITMENU	User selected the extension's menu.	FMEVENT_LOAD	File Manager is loading the extension DLL.	FMEVENT_SELCHANGE	Selection in File Manager's directory window, or Search Results window, changed.	FMEVENT_UNLOAD	File Manager is unloading the extension DLL.	FMEVENT_USER_REFRESH	User chose the Refresh command from the Window menu.
Value	Meaning														
1-99	Identifier for the menu item that the user selected.														
FMEVENT_INITMENU	User selected the extension's menu.														
FMEVENT_LOAD	File Manager is loading the extension DLL.														
FMEVENT_SELCHANGE	Selection in File Manager's directory window, or Search Results window, changed.														
FMEVENT_UNLOAD	File Manager is unloading the extension DLL.														
FMEVENT_USER_REFRESH	User chose the Refresh command from the Window menu.														
lParam	Specifies 32 bits of additional message-dependent information.														

#### Returns

The callback function should return the result of the message processing. The actual return value depends on the message that is processed.

#### Comments

Whenever File Manager calls the FMExtensionProc function, it waits to refresh its directory windows (for changes in the file system) until after the function returns. This allows the extension to perform large numbers of file operations without excessive repainting by the File Manager. The extension does not need to send the FM\_REFRESH\_WINDOWS message to notify File Manager to repaint its windows.

#### See Also

FM\_REFRESH\_WINDOWS, FMS\_LOAD

## UndeleteFile

```
#include <wfext.h>
```

```
int CALLBACK UndeleteFile(HWND hwndParent, LPSTR lpszDir,  
                          /* handle of File Manager window */  
                          /* address of  
name of initial directory  
/
```

The UndeleteFile function is an application-defined callback function that File Manager calls when the user chooses the Undelete command from the File Manager File menu.

Parameter	Description
hwndParent	Identifies the File Manager window. An "undelete" dynamic-link library (DLL) should use this handle to specify the parent window for any dialog box or message box the DLL may display.
lpszDir	Points to a null-terminated string that contains the name of the initial directory.

### Returns

The return value is one of the following, if the function is successful:

Value	Meaning
-1	An error occurred.
IDOK	A file was undeleted. File Manager will repaint its windows.
IDCANCEL	No file was undeleted.

### File Manager Extension Functions (3.1)

EMExtensionProc Processes messages for a File Manager extension

UndeleteFile Processes the File Manager Undelete command

## FMEVENT\_INITMENU

The FMEVENT\_INITMENU message is sent to an extension dynamic-link library (DLL) when the user selects the menu for the extension from File Manager's menu bar. The extension can use this notification to initialize menu items in the menu.

Parameter	Description
lParam	Specifies the menu handle in the high-order word. The low-order word specifies the delta value for the menu item.

### Returns

This message does not return a value.

### Comments

An extension receives this message only when the user selects the top-level menu. If the extension contains submenus, it must initialize them at the same time as the top-level menu.

### See Also

FMEExtensionProc

## FMEVENT\_LOAD

The FMEVENT\_LOAD message is sent to an extension dynamic-link library (DLL) when File Manager is loading the DLL.

Parameter	Description
lParam	Points to an <u>FMS_LOAD</u> structure that specifies the menu-item delta value. An extension DLL should save the menu-item delta value and fill the other structure members with information about the extension.

### Returns

This message does not return a value.

### Comments

An application should fill the dwSize, szMenuName, and hMenu members. It should also save the value of the wMenuDelta member and use it to identify menu items when modifying the menu. For more information, see the description of the FMS\_LOAD structure.

### See Also

FMExtensionProc, FMS\_LOAD

## FMEVENT\_SELCHANGE

The FMEVENT\_SELCHANGE message is sent to an extension dynamic-link library (DLL) when the user selects a filename in File Manager's directory window or Search Results window.

Parameter	Description
IParam	Not used.

### Returns

This message does not return a value.

### Comments

Changes in the tree half of the directory window do not produce this message.

Because the user can change the selection many times, the extension DLL must return promptly after processing this message to avoid slowing the selection process for the user.

### See Also

FMExtensionProc, FMEVENT\_UNLOAD

## FMEVENT\_UNLOAD

The FMEVENT\_UNLOAD message is sent to an extension dynamic-link library (DLL) when File Manager is unloading the DLL.

Parameter	Description
lParam	Not used.

### Returns

This message does not return a value.

### Comments

The hwnd and hMenu values passed with the FMEVENT\_LOAD and FMEVENT\_INITMENU messages may not be valid at the time of this message.

### See Also

FMExtensionProc, FMEVENT\_INITMENU, FMEVENT\_LOAD

## FMEVENT\_USER\_REFRESH

The FMEVENT\_USER\_REFRESH message is sent to an extension dynamic-link library (DLL) when the user invokes File Manager's Refresh command in the Window menu. The extension can use this notification to update its menu.

Parameter	Description
IParam	Not used.

### Returns

This message does not return a value.

### See Also

FMExtensionProc



## FM\_GETDRIVEINFO

A File Manager extension sends an FM\_GETDRIVEINFO message to retrieve drive information from the active File Manager window.

Parameter	Description
wParam	Not used.
lParam	Points to an <u>FMS_GETDRIVEINFO</u> structure that receives drive information.

### Returns

The return value is always nonzero.

### Comments

If a -1 is returned in the dwTotalSpace or dwFreeSpace members of the FMS\_GETDRIVEINFO structure, the extension library must compute the value or values.

### See Also

FMExtensionProc, FMS\_GETDRIVEINFO

## FM\_GETFILESEL

A File Manager extension sends an FM\_GETFILESEL message to retrieve information about a selected file from the active File Manager window (either the directory window or the Search Results window).

Parameter	Description
wParam	Specifies the zero-based index of the selected file to retrieve.
lParam	Points to an <u>FMS_GETFILESEL</u> structure that receives information about the selection.

### Returns

The return value is the zero-based index of the selected file that was retrieved.

### Comments

An extension can use the FM\_GETSELCOUNT message to obtain the count of selected files.

The szName member of the FMS\_GETFILESEL structure consists of an OEM character string. Before displaying this string, an extension should use the OemToAnsi function to convert the string to a Windows ANSI character string. If a string is to be passed to the file system (MS-DOS), an extension should not convert it.

### See Also

FMExtensionProc, FM\_GETFILESELLFN, FM\_GETSELCOUNT, FM\_GETSELCOUNTLFN, OemToAnsi, FMS\_GETFILESEL

## FM\_GETFILESELLFN

A File Manager extension sends an FM\_GETFILESELLFN message to retrieve information about a selected file from the active File Manager window (either the directory window or the Search Results window). The selected file can have a long filename.

Parameter	Description
wParam	Specifies the zero-based index of the selected file to retrieve.
lParam	Points to an <u>FMS_GETFILESEL</u> structure that receives information about the selection.

### Returns

The return value is the zero-based index of the selected file that was retrieved.

### Comments

Only extensions that support long filenames (for example, network-aware extensions) should use this message.

An extension can use the FM\_GETSELCOUNT message to obtain the count of selected files.

The szName member of the FMS\_GETFILESEL structure consists of an OEM character string. Before displaying this string, an extension should use the OemToAnsi function to convert the string to a Windows ANSI character string. If a string is to be passed to the file system (MS-DOS), an extension should not convert it.

### See Also

FMExtensionProc, FM\_GETFILESEL, FM\_GETSELCOUNT, FM\_GETSELCOUNTLFN, OemToAnsi, FMS\_GETFILESEL

## FM\_GETFOCUS

A File Manager extension sends a FM\_GETFOCUS message to retrieve the type of the File Manager window that has the input focus.

Parameter	Description
wParam	Not used.
lParam	Not used.

### Returns

The return value indicates the type of File Manager window that has input focus. It can have one of the following values:

Value	Meaning
FMFOCUS_DIR	Directory portion of a directory window
FMFOCUS_TREE	Tree portion of a directory window
FMFOCUS_DRIVES	Drive bar of a directory window
FMFOCUS_SEARCH	Search Results window

## FM\_GETSELCOUNT

A File Manager extension sends a FM\_GETSELCOUNT message to retrieve a count of the selected files in the directory or the Search Results window, depending on which is the active window.

Parameter	Description
wParam	Not used.
lParam	Not used.

### Returns

The return value is the number of selected files.

### See Also

FM\_GETFILESEL, FM\_GETFILESELLFN, FM\_GETSELCOUNTLFN

## FM\_GETSELCOUNTLFN

A File Manager extension sends an FM\_GETSELCOUNTLFN message to retrieve the number of selected files in the directory or the Search Results window, depending on which is the active window. The count includes files that have long filenames.

Parameter	Description
wParam	Not used.
lParam	Not used.

### Returns

The return value is the number of selected files.

### Comments

Only extensions that support long filenames (for example, network-aware extensions) should use this message.

### See Also

FM\_GETFILESEL, FM\_GETFILESELLFN, FM\_GETSELCOUNT

## FM\_REFRESH\_WINDOWS

A File Manager extension sends an FM\_REFRESH\_WINDOWS message to cause File Manager to repaint either its active window or all of its windows.

Parameter	Description
wParam	Specifies whether File Manager repaints its active window or all of its windows. If this parameter is nonzero, File Manager repaints all of its windows. If this parameter is zero, File Manager repaints only its active window.
lParam	Not used.

### Returns

This message does not return a meaningful value.

### Comments

File system changes caused by an extension are automatically detected by File Manager. An extension should use this message only in situations where drive connections are made or canceled.

### See Also

FMEExtensionProc

## FM\_RELOAD\_EXTENSIONS

A File Manager extension (or another application) sends an FM\_RELOAD\_EXTENSIONS message to cause File Manager to reload all extension dynamic-link libraries (DLLs) listed in the [AddOns] section of the WINFILE.INI file.

Parameter	Description
wParam	Not used.
lParam	Not used.

### Returns

This message does not return a meaningful value.

### Comments

Other applications can use the PostMessage function to send this message to File Manager. To obtain the appropriate File Manager window handle, an application can specify "WFS\_Frame" as the lpszClassName parameter in a call to the FindWindow function.

### See Also

FindWindow, FMExtensionProc, PostMessage



### File Manager Extension Messages (3.1)

EMEVENT\_INITMENU

EMEVENT\_LOAD

EMEVENT\_SELCHANGE

EMEVENT\_UNLOAD

EMEVENT\_USER\_REFRESH

EM\_GETDRIVEINFO

Retrieves drive data from active window

EM\_GETFILESEL

Retrieves data about a selected file

EM\_GETFILESELLEN

Retrieves data about a selected file

EM\_GETFOCUS

Retrieves the type of the File Manager focus window

EM\_GETSELCOUNT

Retrieves the count of selected files

EM\_GETSELCOUNTLEN

Retrieves the count of selected files

EM\_REFRESH\_WINDOWS

Repaints File Manager's windows

EM\_RELOAD\_EXTENSIONS

Reloads File Manager extension DLLs

## AbortDoc (3.1)

```
int AbortDoc(hdc)
HDC hdc; /* handle of device context */
```

The AbortDoc function terminates the current print job and erases everything drawn since the last call to the StartDoc function. This function replaces the ABORTDOC printer escape for Windows version 3.1.

Parameter	Description
hdc	Identifies the device context for the print job.

### Returns

The return value is greater than or equal to zero if the function is successful. Otherwise, it is less than zero.

### Comments

Applications should call the AbortDoc function to terminate a print job because of an error or if the user chooses to cancel the job. To end a successful print job, an application should use the EndDoc function.

If Print Manager was used to start the print job, calling the AbortDoc function erases the entire spool job--the printer receives nothing. If Print Manager was not used to start the print job, the data may have been sent to the printer before AbortDoc was called. In this case, the printer driver would have reset the printer (when possible) and closed the print job.

### See Also

EndDoc, SetAbortProc, StartDoc

## AddFontResource (2.x)

```
int AddFontResource(lpszFilename)
LPCSTR lpszFilename;          /* address of filename */
```

The AddFontResource function adds a font resource to the Windows font table. Any application can then use the font.

Parameter	Description
lpszFilename	Points to a character string that names the font resource file or that contains a handle of a loaded module. If this parameter points to a font resource filename, it must be a valid MS-DOS filename, including an extension, and the string must be null-terminated. The system passes this string to the <u>LoadLibrary</u> function if the font resource must be loaded.

### Returns

The return value specifies the number of fonts added if the function is successful. Otherwise, it is zero.

### Comments

Any application that adds or removes fonts from the Windows font table should send a WM\_FONTCHANGE message to all top-level windows in the system by using the SendMessage function with the hwnd parameter set to 0xFFFF.

When font resources added by using AddFontResource are no longer needed, you should remove them by using the RemoveFontResource function.

### Example

The following example uses the AddFontResource function to add a font resource from a file, notifies other applications by using the SendMessage function, then removes the font resource by using the RemoveFontResource function:

```
AddFontResource("fontres.fon");
SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
. /* Work with the font. */
.
if (RemoveFontResource("fontres.fon")) {
    SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
    return TRUE;
}
else
    return FALSE;
```

### See Also

LoadLibrary, RemoveFontResource, SendMessage

## AnimatePalette (3.0)

```
void AnimatePalette(hpal, iStart, cEntries, lppe)
HPALETTE hpal; /* handle of palette */
UINT iStart; /* first palette entry to animate */
/
UINT cEntries; /* number of entries in palette */
/
const PALETTEENTRY FAR* lppe; /* address of color structure */
/
```

The AnimatePalette function replaces entries in the specified logical palette. An application does not have to update the client area when it calls AnimatePalette, because Windows maps the new entries into the system palette immediately.

Parameter	Description
hpal	Identifies the logical palette.
iStart	Specifies the first entry in the palette to be animated.
cEntries	Specifies the number of entries in the palette to be animated.
lppe	Points to the first member of an array of PALETTEENTRY structures. These palette entries will replace the palette entries identified by the iStart and cEntries parameters.

### Returns

This function does not return a value.

### Comments

The AnimatePalette function can change an entry in a logical palette only when the PC\_RESERVED flag is set in the corresponding palPaletteEntry member of the LOGPALETTE structure that defines the current logical palette.

### Example

The following example initializes a LOGPALETTE structure and an array of PALETTEENTRY structures, uses the CreatePalette function to retrieve a handle of a logical palette, and then uses the AnimatePalette function to map the entries into the system palette:

```
#define NUMENTRIES 128
HPALETTE hpal;
PALETTEENTRY ape[NUMENTRIES];

plgpl = (LOGPALETTE*) LocalAlloc(LPTR,
    sizeof(LOGPALETTE) + cColors * sizeof(PALETTEENTRY));

plgpl->palNumEntries = cColors;
plgpl->palVersion = 0x300;

for (i = 0, red = 0, green = 127, blue = 127; i < NUMENTRIES;
    i++, red += 1, green += 1, blue += 1) {
    ape[i].peRed =
    plgpl->palPalEntry[i].peRed = LOBYTE(red);
    ape[i].peGreen =
    plgpl->palPalEntry[i].peGreen = LOBYTE(green);
    ape[i].peBlue =
    plgpl->palPalEntry[i].peBlue = LOBYTE(blue);
    ape[i].peFlags =
    plgpl->palPalEntry[i].peFlags = PC_RESERVED;
}
hpal = CreatePalette(plgpl);
LocalFree((HLOCAL) plgpl);
AnimatePalette(hpal, 0, NUMENTRIES, (PALETTEENTRY FAR*) &ape);
```

See Also

CreatePalette, LOGPALETTE, PALETTEENTRY

## Arc (2.x)

```
BOOL Arc(hdc, nLeftRect, nTopRect, nRightRect, nBottomRect, nXStartArc, nYStartArc, nXEndArc,  
         nYEndArc)
```

```
HDC hdc;          /* handle of device context          */  
int nLeftRect;   /* x-coordinate upper-left corner bounding rectangle  */  
/   
int nTopRect;   /* y-coordinate upper-left corner bounding rectangle  */  
/   
int nRightRect; /* x-coordinate lower-right corner bounding rectangle */  
/   
int nBottomRect; /* y-coordinate lower-right corner bounding rectangle */  
/   
int nXStartArc; /* x-coordinate arc starting point                   */  
/   
int nYStartArc; /* y-coordinate arc starting point                   */  
/   
int nXEndArc;   /* x-coordinate arc ending point                     */  
/   
int nYEndArc;   /* y-coordinate arc ending point                     */  
/
```

The Arc function draws an elliptical arc.

Parameter	Description
hdc	Identifies the device context.
nLeftRect	Specifies the logical x-coordinate of the upper-left corner of the bounding rectangle.
nTopRect	Specifies the logical y-coordinate of the upper-left corner of the bounding rectangle.
nRightRect	Specifies the logical x-coordinate of the lower-right corner of the bounding rectangle.
nBottomRect	Specifies the logical y-coordinate of the lower-right corner of the bounding rectangle.
nXStartArc	Specifies the logical x-coordinate of the point that defines the arc's starting point. This point need not lie exactly on the arc.
nYStartArc	Specifies the logical y-coordinate of the point that defines the arc's starting point. This point need not lie exactly on the arc.
nXEndArc	Specifies the logical x-coordinate of the point that defines the arc's endpoint. This point need not lie exactly on the arc.
nYEndArc	Specifies the logical y-coordinate of the point that defines the arc's endpoint. This point need not lie exactly on the arc.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The arc drawn by using the Arc function is a segment of the ellipse defined by the specified bounding rectangle. The starting point of the arc is the point at which a ray drawn from the center of the bounding rectangle through the specified starting point intersects the ellipse. The end point of the arc is the point at which a ray drawn from the center of the bounding rectangle through the specified end point intersects the ellipse. The arc is drawn in a counterclockwise direction. Since an arc is not a closed figure, it is not filled.

Both the width and the height of a rectangle must be greater than 2 units and less than 32,767 units.

### Example

The following example uses a RECT structure to store the points defining the bounding rectangle and uses POINT structures to store the coordinates that specify the beginning and end of the arc:

HDC hdc;

**RECT** rc = { 10, 10, 180, 140 };

**POINT** ptStart = { 12, 12 };

**POINT** ptEnd = { 128, 135 };

Arc(hdc, rc.left, rc.top, rc.right, rc.bottom,  
ptStart.x, ptStart.y, ptEnd.x, ptEnd.y);

See Also

Chord, POINT, RECT

## BitBlt (2.x)

```
BOOL BitBlt(hdcDest, nXDest, nYDest, nWidth, nHeight, hdcSrc, nXSrc, nYSrc, dwRop)
HDC hdcDest;          /* handle of destination device context */
int nXDest;           /* upper-left
corner destination rectangle
/
int nYDest;
* upper-left corner destination rectangle
/
int nWidth;
* bitmap width
/
int nHeight;
* bitmap height
/
HDC hdcSrc;
* handle of source device context
/
int nXSrc;
* upper-left corner source bitmap
/
int nYSrc;
* upper-left corner source bitmap
/
DWORD dwRop;
* raster operation for copy
/
```

The BitBlt function copies a bitmap from a specified device context to a destination device context.

Parameter	Description
hdcDest	Identifies the destination device context.
nXDest	Specifies the logical x-coordinate of the upper-left corner of the destination rectangle.
nYDest	Specifies the logical y-coordinate of the upper-left corner of the destination rectangle.
nWidth	Specifies the width, in logical units, of the destination rectangle and source bitmap.
nHeight	Specifies the height, in logical units, of the destination rectangle and source bitmap.
hdcSrc	Identifies the device context from which the bitmap will be copied. This parameter must be NULL if the dwRop parameter specifies a raster operation that does not include a source. This parameter can specify a memory device context.
nXSrc	Specifies the logical x-coordinate of the upper-left corner of the source bitmap.
nYSrc	Specifies the logical y-coordinate of the upper-left corner of the source bitmap.
dwRop	Specifies the raster operation to be performed. Raster operation codes define how the graphics device interface (GDI) combines colors in output operations that involve a current brush, a possible source bitmap, and a destination bitmap. This parameter can be one of the following:

Code	Description
<b>BLACKNESS</b>	Turns all output black.
<b>DSTINVERT</b>	Inverts the destination bitmap.
<b>MERGECOPY</b>	Combines the pattern and the source bitmap by using the Boolean AND operator.
<b>MERGEPAINT</b>	Combines the inverted source bitmap with the destination bitmap by using the Boolean OR operator.
<b>NOTSRCCOPY</b>	Copies the inverted source bitmap to the destination.
<b>NOTSRCERASE</b>	Inverts the result of combining the destination and source bitmaps by using the Boolean OR operator.
<b>PATCOPY</b>	Copies the pattern to the destination bitmap.
<b>PATINVERT</b>	Combines the destination bitmap with the pattern by using the Boolean XOR operator.



<u>PATPAINT</u>	Combines the inverted source bitmap with the pattern by using the Boolean OR operator. Combines the result of this operation with the destination bitmap by using the Boolean OR operator.
<u>SRCAND</u>	Combines pixels of the destination and source bitmaps by using the Boolean AND operator.
<u>SRCCOPY</u>	Copies the source bitmap to the destination bitmap.
<u>SRCERASE</u>	Inverts the destination bitmap and combines the result with the source bitmap by using the Boolean AND operator.
<u>SRCINVERT</u>	Combines pixels of the destination and source bitmaps by using the Boolean XOR operator.
<u>SRCPAINT</u>	Combines pixels of the destination and source bitmaps by using the Boolean OR operator.
<u>WHITENESS</u>	Turns all output white.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

An application that uses the BitBlt function to copy pixels from one window to another window or from a source rectangle in a window into a target rectangle in the same window should set the CS\_BYTEALIGNWINDOW or CS\_BYTEALIGNCLIENT flag when registering the window classes. By aligning the windows or client areas on byte boundaries, the application can ensure that the BitBlt operations occur on byte-aligned rectangles. BitBlt operations on byte-aligned rectangles are considerably faster than BitBlt operations on rectangles that are not byte-aligned.

GDI transforms the nWidth and nHeight parameters, once by using the destination device context, and once by using the source device context. If the resulting extents do not match, GDI uses the StretchBlt function to compress or stretch the source bitmap as necessary. If destination, source, and pattern bitmaps do not have the same color format, the BitBlt function converts the source and pattern bitmaps to match the destination. The foreground and background colors of the destination bitmap are used in the conversion.

When the BitBlt function converts a monochrome bitmap to color, it sets white bits (1) to the background color and black bits (0) to the foreground color. The foreground and background colors of the destination device context are used. To convert color to monochrome, BitBlt sets pixels that match the background color to white and sets all other pixels to black. BitBlt uses the foreground and background colors of the source (color) device context to convert from color to monochrome.

The foreground color is the current text color for the specified device context, and the background color is the current background color for the specified device context.

Not all devices support the BitBlt function. An application can determine whether a device supports BitBlt by calling the GetDeviceCaps function and specifying the RASTERCAPS index.

#### Example

The following example loads a bitmap, retrieves its dimensions, and displays it in a window:

```
HDC hdc, hdcMemory;
HBITMAP hbmpMyBitmap, hbmpOld;
BITMAP bm;

hbmpMyBitmap = LoadBitmap(hinst, "MyBitmap");
GetObject(hbmpMyBitmap, sizeof(BITMAP), &bm);

hdc = GetDC(hwnd);
hdcMemory = CreateCompatibleDC(hdc);
hbmpOld = SelectObject(hdcMemory, hbmpMyBitmap);

BitBlt(hdc, 0, 0, bm.bmWidth, bm.bmHeight, hdcMemory, 0, 0, SRCCOPY);
SelectObject(hdcMemory, hbmpOld);

DeleteDC(hdcMemory);
ReleaseDC(hwnd, hdc);
```

See Also

GetDeviceCaps, PatBlt, SetTextColor, StretchBlt, StretchDIBits

BLACKNESS 0x00000042L

Turns all output black.

BLACKNESS 0x00000042L

DSTINVERT 0x00550009L

Inverts the destination bitmap.

DSTINVERT 0x00550009L

MERGECOPY 0x00C000CAL

Combines the pattern and the source bitmap by using the Boolean AND operator.

MERGECOPY 0x00C000CAL



MERGEPAINT 0x00BB0226L

Combines the inverted source bitmap with the destination bitmap by using the Boolean OR operator.

MERGEPAINT 0x00BB0226L

NOTSRCCOPY 0x00330008L

Copies the inverted source bitmap to the destination.

NOTSRCCOPY 0x00330008L

NOTSRCERASE 0x001100A6L

Inverts the result of combining the destination and source bitmaps by using the Boolean OR operator.

NOTSRCERASE 0x001100A6L

PATCOPY 0x00F00021L

Copies the pattern to the destination bitmap.

PATCOPY 0x00F00021L



PATINVERT 0x005A0049L

Combines the destination bitmap with the pattern by using the Boolean XOR operator.

PATINVERT 0x005A0049L

PATPAINT 0x00FB0A09L

Combines the inverted source bitmap with the pattern by using the Boolean OR operator. Combines the result of this operation with the destination bitmap by using the Boolean OR operator.

PATPAINT 0x00FB0A09L

SRCAND 0x008800C6L

Combines pixels of the destination and source bitmaps by using the Boolean AND operator.

SRCAND 0x008800C6L

SRCCOPY 0x00CC0020L

Copies the source bitmap to the destination bitmap.

SRCCOPY 0x00CC0020L



SRCERASE 0x00440328L

Inverts the destination bitmap and combines the result with the source bitmap by using the Boolean AND operator.

SRCERASE 0x00440328L

SRCINVERT 0x00660046L

Combines pixels of the destination and source bitmaps by using the Boolean XOR operator.

SRCINVERT 0x00660046L

SRCPAINT 0x00EE0086L

Combines pixels of the destination and source bitmaps by using the Boolean OR operator.

SRCPAINT 0x00EE0086L

WHITENESS 0x00FF0062L

Turns all output white.

WHITENESS 0x00FF0062L



## Chord (2.x)

```
BOOL Chord(hdc, nLeftRect, nTopRect, nRightRect, nBottomRect, nXStartLine, nYStartLine,  
           nXEndLine, nYEndLine)  
HDC hdc;          /* handle of device context */  
int nLeftRect;  
/* x-coordinate upper-left corner bounding rectangle  
/  
int nTopRect;  
/* y-coordinate upper-left corner bounding rectangle  
/  
int nRightRect;  
/* x-coordinate lower-right corner bounding rectangle  
/  
int nBottomRect;  
/* y-coordinate lower-right corner bounding rectangle  
/  
int nXStartLine;  
/* x-coordinate line-segment starting point  
/  
int nYStartLine;  
/* y-coordinate line-segment starting point  
/  
int nXEndLine;  
/* x-coordinate line-segment ending point  
/  
int nYEndLine;  
/* y-coordinate line-segment ending point  
/
```

The Chord function draws a chord (a closed figure bounded by the intersection of an ellipse and a line segment).

Parameter	Description
hdc	Identifies the device context.
nLeftRect	Specifies the logical x-coordinate of the upper-left corner of the bounding rectangle.
nTopRect	Specifies the logical y-coordinate of the upper-left corner of the bounding rectangle.
nRightRect	Specifies the logical x-coordinate of the lower-right corner of the bounding rectangle.
nBottomRect	Specifies the logical y-coordinate of the lower-right corner of the bounding rectangle.
nXStartLine	Specifies the logical x-coordinate of the starting point of the line segment.
nYStartLine	Specifies the logical y-coordinate of the starting point of the line segment.
nXEndLine	Specifies the logical x-coordinate of the ending point of the line segment.
nYEndLine	Specifies the logical y-coordinate of the ending point of the line segment.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The (nLeftRect, nTopRect) and (nRightRect, nBottomRect) parameter combinations specify the upper-left and lower-right corners, respectively, of a rectangle bounding the ellipse that is part of the chord. The (nXStartLine, nYStartLine) and (nXEndLine, nYEndLine) parameter combinations specify the endpoints of a line that intersects the ellipse. The chord is drawn by using the selected pen and is filled by using the selected brush.

The figure the Chord function draws extends up to but does not include the right and bottom coordinates. This means that the height of the figure is determined as follows:

$nBottomRect - nTopRect$

The width of the figure is determined similarly:

$nRightRect - nLeftRect$

### Example

The following example uses a RECT structure to store the points defining the bounding rectangle and uses POINT structures to store the coordinates that specify the beginning and end of the chord:

```
HDC hdc;
```

```
RECT rc = { 10, 10, 180, 140 };
```

```
POINT ptStart = { 12, 12 };
```

```
POINT ptEnd = { 128, 135 };
```

```
Chord(hdc, rc.left, rc.top, rc.right, rc.bottom,  
      ptStart.x, ptStart.y, ptEnd.x, ptEnd.y);
```

See Also

Arc, POINT, RECT

## CloseMetaFile (2.x)

```
HMETAFILE CloseMetaFile(hdc)
HDC hdc; /* handle of device context */
```

The CloseMetaFile function closes a metafile device context and creates a handle of a metafile. An application can use this handle to play the metafile.

Parameter	Description
hdc	Identifies the metafile device context to be closed.

### Returns

The return value is the handle of the metafile if the function is successful. Otherwise, it is NULL.

### Comments

If a metafile handle created by using the CloseMetaFile function is no longer needed, you should remove it (using the [DeleteMetaFile](#) function).

### Example

The following example creates a device-context handle of a memory metafile, draws a line in the device context, retrieves a handle of the metafile, plays the metafile, and finally deletes the metafile.

```
HDC hdcMeta;
HMETAFILE hmf;
```

```
hdcMeta = CreateMetaFile(NULL);
MoveTo(hdcMeta, 10, 10);
LineTo(hdcMeta, 100, 100);
hmf = CloseMetaFile(hdcMeta);
PlayMetaFile(hdc, hmf);
DeleteMetaFile(hmf);
```

### See Also

[CreateMetaFile](#), [DeleteMetaFile](#), [PlayMetaFile](#)

## CombineRgn (2.x)

```
int CombineRgn(hrgnDest, hrgnSrc1, hrgnSrc2, fCombineMode)
HRGN hrgnDest;          /* handle of region to receive combined regions */
HRGN hrgnSrc1;
* handle of first source region
/
HRGN hrgnSrc2;
* handle of second source region
/
int fCombineMode;
* mode for combining regions
/
```

The CombineRgn function creates a new region by combining two existing regions.

Parameter	Description												
hrgnDest	Identifies an existing region that will be replaced by the new region.												
hrgnSrc1	Identifies an existing region.												
hrgnSrc2	Identifies an existing region.												
fCombineMode	Specifies the operation to use when combining the two source regions. This parameter can be any one of the following values:												
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>RGN_AND</u></td><td>Uses overlapping areas of both regions (intersection).</td></tr><tr><td><u>RGN_COPY</u></td><td>Creates a copy of region 1 (identified by the hrgnSrc1 parameter).</td></tr><tr><td><u>RGN_DIFF</u></td><td>Creates a region consisting of the areas of region 1 (identified by hrgnSrc1) that are not part of region 2 (identified by the hrgnSrc2 parameter).</td></tr><tr><td><u>RGN_OR</u></td><td>Combines all of both regions (union).</td></tr><tr><td><u>RGN_XOR</u></td><td>Combines both regions but removes overlapping areas.</td></tr></tbody></table>	Value	Meaning	<u>RGN_AND</u>	Uses overlapping areas of both regions (intersection).	<u>RGN_COPY</u>	Creates a copy of region 1 (identified by the hrgnSrc1 parameter).	<u>RGN_DIFF</u>	Creates a region consisting of the areas of region 1 (identified by hrgnSrc1) that are not part of region 2 (identified by the hrgnSrc2 parameter).	<u>RGN_OR</u>	Combines all of both regions (union).	<u>RGN_XOR</u>	Combines both regions but removes overlapping areas.
Value	Meaning												
<u>RGN_AND</u>	Uses overlapping areas of both regions (intersection).												
<u>RGN_COPY</u>	Creates a copy of region 1 (identified by the hrgnSrc1 parameter).												
<u>RGN_DIFF</u>	Creates a region consisting of the areas of region 1 (identified by hrgnSrc1) that are not part of region 2 (identified by the hrgnSrc2 parameter).												
<u>RGN_OR</u>	Combines all of both regions (union).												
<u>RGN_XOR</u>	Combines both regions but removes overlapping areas.												

### Returns

The return value specifies that the resulting region has overlapping borders (COMPLEXREGION), is empty (NULLREGION), or has no overlapping borders (SIMPLEREGION), if the function is successful. Otherwise, the return value is ERROR.

### Comments

The size of a region is limited to 32,000 by 32,000 logical units or 64K of memory, whichever is smaller.

The CombineRgn function replaces the region identified by the hrgnDest parameter with the combined region. To use CombineRgn most efficiently, hrgnDest should be a trivial region, as shown in the following example.

### Example

The following example creates two source regions and an empty destination region, uses the CombineRgn function to create a complex region, selects the region into a device context, and then uses the PaintRgn function to display the region:

```
HDC hdc;
HRGN hrgnDest, hrgnSrc1, hrgnSrc2;

hrgnDest = CreateRectRgn(0, 0, 0, 0);
hrgnSrc1 = CreateRectRgn(10, 10, 110, 110);
hrgnSrc2 = CreateRectRgn(90, 90, 200, 150);

CombineRgn(hrgnDest, hrgnSrc1, hrgnSrc2, RGN_OR);
SelectObject(hdc, hrgnDest);
PaintRgn(hdc, hrgnDest);
```

### See Also

CreateRectRgn, PaintRgn

RGN\_AND 1

Uses overlapping areas of both regions (intersection).

RGN\_AND 1

RGN\_COPY 5

Creates a copy of region 1 (identified by the hrgnSrc1 parameter).

RGN\_COPY 5



RGN\_DIFF 4

Creates a region consisting of the areas of region 1 (identified by hrgnSrc1) that are not part of region 2 (identified by the hrgnSrc2 parameter).

RGN\_DIFF 4

RGN\_OR 2

Combines all of both regions (union).

RGN\_OR 2

RGN\_XOR 3

Combines both regions but removes overlapping areas.

RGN\_XOR 3

## CopyMetaFile (2.x)

```
HMETAFILE CopyMetaFile(hmfSrc, lpszFile)
HMETAFILE hmfSrc;          /* handle of metafile to copy          */
LPCSTR lpszFile;           /* address of name of copied metafile */
/
```

The CopyMetaFile function copies a source metafile to a specified file and returns a handle of the new metafile.

Parameter	Description
hmfSrc	Identifies the source metafile to be copied.
lpszFile	Points to a null-terminated string that specifies the filename of the copied metafile. If this value is NULL, the source metafile is copied to a memory metafile.

### Returns

The return value is the handle of the new metafile if the function is successful. Otherwise, it is NULL.

### Example

The following example copies a metafile to a specified file, plays the copied metafile, retrieves a handle of the copied metafile, changes the position at which the metafile is played 200 logical units to the right, and then plays the metafile at the new location:

```
HANDLE hmf, hmfSource, hmfOld;
LPSTR lpszFile1 = "MFTest";

hmf = CopyMetaFile(hmfSource, lpszFile1);
PlayMetaFile(hdc, hmf);
DeleteMetaFile(hmf);

hmfOld = GetMetaFile(lpszFile1);
SetWindowOrg(hdc, -200, 0);
PlayMetaFile(hdc, hmfOld);

DeleteMetaFile(hmfSource);
DeleteMetaFile(hmfOld);
```

See Also  
[GetMetaFile](#), [PlayMetaFile](#), [SetWindowOrg](#)

## CreateBitmap (2.x)

```
HBITMAP CreateBitmap(nWidth, nHeight, cbPlanes, cbBits, lpvBits)
int nWidth;          /* bitmap width */
int nHeight;         /* bitmap height */
height
/
UINT cbPlanes;
* number of color planes
/
UINT cbBits;
* number of bits per pixel
/
const void FAR* lpvBits;
* address of array with bitmap bits
/
```

The CreateBitmap function creates a device-dependent memory bitmap that has the specified width, height, and bit pattern.

Parameter	Description
nWidth	Specifies the width, in pixels, of the bitmap.
nHeight	Specifies the height, in pixels, of the bitmap.
cbPlanes	Specifies the number of color planes in the bitmap. The number of bits per plane is the product of the plane's width, height, and bits per pixel (nWidth nHeight cbBits).
cbBits	Specifies the number of color bits per display pixel.
lpvBits	Points to an array of short integers that contains the initial bitmap bit values. If this parameter is NULL, the new bitmap is left uninitialized.

### Returns

The return value is the handle of the bitmap if the function is successful. Otherwise, it is NULL.

### Comments

The bitmap created by the CreateBitmap function can be selected as the current bitmap for a memory device context by using the [SelectObject](#) function.

For a color bitmap, either the cbPlanes or cbBits parameter should be set to 1. If both of these parameters are set to 1, CreateBitmap creates a monochrome bitmap.

Although a bitmap cannot be copied directly to a display device, the [BitBlt](#) function can copy it from a memory device context (in which it is the current bitmap) to any compatible device context, including a screen device context.

When it has finished using a bitmap created by CreateBitmap, an application should select the bitmap out of the device context and then remove the bitmap by using the [DeleteObject](#) function.

### Example

The following example uses the CreateBitmap function to create a bitmap with a zigzag pattern and then uses the [PatBlt](#) function to fill the client area with that pattern:

```
HDC hdc;
HBITMAP hbmp;
HBRUSH hbr, hbrPrevious;
RECT rc;

int aZigzag[] = { 0xFF, 0xF7, 0xEB, 0xDD, 0xBE, 0x7F, 0xFF, 0xFF };

hbmp = CreateBitmap(8, 8, 1, 1, aZigzag);
hbr = CreatePatternBrush(hbmp);

hdc = GetDC(hwnd);
UnrealizeObject(hbr);
hbrPrevious = SelectObject(hdc, hbr);
GetClientRect(hwnd, &rc);
```



```
PatBlt(hdc, rc.left, rc.top,  
        rc.right - rc.left, rc.bottom - rc.top, PATCOPY);  
SelectObject(hdc, hbrPrevious);  
ReleaseDC(hwnd, hdc);
```

```
DeleteObject(hbr);  
DeleteObject(hbmp);
```

See Also

BitBlt, CreateBitmapIndirect, CreateCompatibleBitmap, CreateDIBitmap, CreateDiscardableBitmap,  
DeleteObject, SelectObject

## CreateBitmapIndirect (2.x)

```
HBITMAP CreateBitmapIndirect(lpbm)
BITMAP FAR* lpbm;          /* address of structure with bitmap information */
```

The `CreateBitmapIndirect` function creates a bitmap that has the width, height, and bit pattern specified in a `BITMAP` structure.

Parameter	Description
<code>lpbm</code>	Points to a <code>BITMAP</code> structure that contains information about the bitmap.

### Returns

The return value is the handle of the bitmap if the function is successful. Otherwise, it is `NULL`.

### Comments

Large bitmaps cannot be displayed on a display device by copying them directly to the device context for that device. Instead, applications should create a memory device context that is compatible with the display device, select the bitmap as the current bitmap for the memory device context, and then use a function such as `BitBlt` or `StretchBlt` to copy it from the memory device context to the display device context. (The `PatBlt` function can copy the bitmap for the current brush directly to the display device context.)

When an application has finished using the bitmap created by the `CreateBitmapIndirect` function, it should select the bitmap out of the device context and then delete the bitmap by using the `DeleteObject` function.

If the `BITMAP` structure pointed to by the `lpbm` parameter has been filled in by using the `GetObject` function, the bits of the bitmap are not specified, and the bitmap is uninitialized. To initialize the bitmap, an application can use a function such as `BitBlt` or `SetDIBits` to copy the bits from the bitmap identified by the first parameter of `GetObject` to the bitmap created by `CreateBitmapIndirect`.

### Example

The following example assigns values to the members of a `BITMAP` structure and then calls the `CreateBitmapIndirect` function to create a bitmap handle:

```
BITMAP bm;
HBITMAP hbm;

int aZigzag[] = { 0xFF, 0xF7, 0xEB, 0xDD, 0xBE, 0x7F, 0xFF, 0xFF };

bm.bmType = 0;
bm.bmWidth = 8;
bm.bmHeight = 8;
bm.bmWidthBytes = 2;
bm.bmPlanes = 1;
bm.bmBitsPixel = 1;
bm.bmBits = aZigzag;

hbm = CreateBitmapIndirect(&bm);
```

### See Also

[BitBlt](#), [CreateBitmap](#), [CreateCompatibleBitmap](#), [CreateDIBitmap](#), [CreateDiscardableBitmap](#), [DeleteObject](#), [GetObject](#), [BITMAP](#)

## CreateBrushIndirect (2.x)

```
HBRUSH CreateBrushIndirect(lplb)
LOGBRUSH FAR* lplb;          /* address of structure with brush attributes */
```

The `CreateBrushIndirect` function creates a brush that has the style, color, and pattern specified in a LOGBRUSH structure. The brush can subsequently be selected as the current brush for any device.

Parameter	Description
<code>lplb</code>	Points to a <u>LOGBRUSH</u> structure that contains information about the brush.

### Returns

The return value is the handle of the brush if the function is successful. Otherwise, it is NULL.

### Comments

A brush created by using a monochrome (one plane, one bit per pixel) bitmap is drawn by using the current text and background colors. Pixels represented by a bit set to 0 are drawn with the current text color, and pixels represented by a bit set to 1 are drawn with the current background color.

When it has finished using a brush created by `CreateBrushIndirect`, an application should select the brush out of the device context in which it was used and then remove the brush by using the DeleteObject function.

### Example

The following example creates a hatched brush with red diagonal hatch marks and uses that brush to fill a rectangle:

```
LOGBRUSH lb;
HBRUSH hbr, hbrOld;

lb.lbStyle = BS_HATCHED;
lb.lbColor = RGB(255, 0, 0);
lb.lbHatch = HS_BDIAGONAL;

hbr = CreateBrushIndirect(&lb);
hbrOld = SelectObject(hdc, hbr);
Rectangle(hdc, 0, 0, 100, 100);
```

### See Also

CreateDIBPatternBrush, CreatePatternBrush, CreateSolidBrush, DeleteObject, GetStockObject, SelectObject, LOGBRUSH, RGB

## CreateCompatibleBitmap (2.x)

```
HBITMAP CreateCompatibleBitmap(hdc, nWidth, nHeight)
HDC hdc; /* handle of device context */
int nWidth; /* bitmap width */
int nHeight; /*
bitmap height */
/ *
```

The CreateCompatibleBitmap function creates a bitmap that is compatible with the given device.

Parameter	Description
hdc	Identifies the device context.
nWidth	Specifies the width, in bits, of the bitmap.
nHeight	Specifies the height, in bits, of the bitmap.

### Returns

The return value is the handle of the bitmap if the function is successful. Otherwise, it is NULL.

### Comments

The bitmap created by the CreateCompatibleBitmap function has the same number of color planes or the same bits-per-pixel format as the given device. It can be selected as the current bitmap for any memory device that is compatible with the one identified by hdc.

If hdc identifies a memory device context, the bitmap returned has the same format as the currently selected bitmap in that device context. A memory device context is a memory object that represents a screen surface. It can be used to prepare images in memory before copying them to the screen surface of the compatible device.

When a memory device context is created, the graphics device interface (GDI) automatically selects a monochrome stock bitmap for it.

Since a color memory device context can have either color or monochrome bitmaps selected, the format of the bitmap returned by the CreateCompatibleBitmap function is not always the same; however, the format of a compatible bitmap for a non-memory device context is always in the format of the device.

When it has finished using a bitmap created by CreateCompatibleBitmap, an application should select the bitmap out of the device context and then remove the bitmap by using the DeleteObject function.

### Example

The following example shows a function named DuplicateBitmap that accepts the handle of a bitmap, duplicates the bitmap, and returns a handle of the duplicate. This function uses the CreateCompatibleDC function to create source and destination device contexts and then uses the GetObject function to retrieve the dimensions of the source bitmap. The CreateCompatibleBitmap function uses these dimensions to create a new bitmap. When each bitmap has been selected into a device context, the BitBlt function copies the bits from the source bitmap to the new bitmap. (Although an application could use the GetDIBits and SetDIBits functions to duplicate a bitmap, the method illustrated in this example is much faster.)

```
HBITMAP PASCAL DuplicateBitmap(HBITMAP hbmpSrc)
{
    HBITMAP hbmpOldSrc, hbmpOldDest, hbmpNew;
    HDC hdcSrc, hdcDest;
    BITMAP bmp;

    hdcSrc = CreateCompatibleDC(NULL);
    hdcDest = CreateCompatibleDC(hdcSrc);

    GetObject(hbmpSrc, sizeof(BITMAP), &bmp);

    hbmpOldSrc = SelectObject(hdcSrc, hbmpSrc);

    hbmpNew = CreateCompatibleBitmap(hdcSrc, bmp.bmWidth,
    bmp.bmHeight);

    hbmpOldDest = SelectObject(hdcDest, hbmpNew);
```

```
BitBlt(hdcDest, 0, 0, bmp.bmWidth, bmp.bmHeight, hdcSrc, 0, 0,  
SRCCOPY);  
  
SelectObject(hdcDest, hbmpOldDest);  
SelectObject(hdcSrc, hbmpOldSrc);  
  
DeleteDC(hdcDest);  
DeleteDC(hdcSrc);  
  
return hbmpNew;  
}
```

See Also

[CreateBitmap](#), [CreateBitmapIndirect](#), [CreateDIBitmap](#), [DeleteObject](#)

## CreateCompatibleDC (2.x)

```
HDC CreateCompatibleDC(hdc)
HDC hdc; /* handle of device context */
```

The CreateCompatibleDC function creates a memory device context that is compatible with the given device.

An application must select a bitmap into a memory device context to represent a screen surface. The device context can then be used to prepare images in memory before copying them to the screen surface of the compatible device.

Parameter	Description
hdc	Identifies the device context. If this parameter is NULL, the function creates a memory device context that is compatible with the system screen.

### Returns

The return value is the handle of the new memory device context if the function is successful. Otherwise, it is NULL.

### Comments

The CreateCompatibleDC function can be used only to create compatible device contexts for devices that support raster operations. To determine whether a device supports raster operations, an application can call the GetDeviceCaps function with the RC\_BITBLT index.

GDI output functions can be used with a memory device context only if a bitmap has been created and selected into that context.

When it has finished using a device context created by CreateCompatibleDC, an application should free the device context by calling the DeleteDC function. All objects selected into the device context after it was created should be selected out and replaced with the original objects before the device context is removed.

### Example

The following example loads a bitmap named Dog, uses the CreateCompatibleDC function to create a memory device context that is compatible with the screen, selects the bitmap into the memory device context, and then uses the BitBlt function to move the bitmap from the memory device context to the screen device context:

```
HDC hdc, hdcMemory;
HBITMAP hbmpMyBitmap, hbmpOld;
BITMAP bm;

hbmpMyBitmap = LoadBitmap(hinst, "MyBitmap");
GetObject(hbmpMyBitmap, sizeof(BITMAP), &bm);

hdc = GetDC(hwnd);
hdcMemory = CreateCompatibleDC(hdc);
hbmpOld = SelectObject(hdcMemory, hbmpMyBitmap);

BitBlt(hdc, 0, 0, bm.bmWidth, bm.bmHeight, hdcMemory, 0, 0, SRCCOPY);
SelectObject(hdcMemory, hbmpOld);

DeleteDC(hdcMemory);
ReleaseDC(hwnd, hdc);
```

See Also  
DeleteDC, GetDeviceCaps

## CreateDC (2.x)

```
#include <print.h>
```

```
HDC CreateDC(lpszDriver, lpszDevice, lpszOutput, lpvInitData)
LPCSTR lpszDriver;          /* address of driver name          */
LPCSTR lpszDevice;         /* address of device name         */
/* address of device name
/
LPCSTR lpszOutput;
/* address of filename or port name
/
const void FAR* lpvInitData;
/* address of initialization data
/
```

The CreateDC function creates a device context for the given device.

Parameter	Description
lpszDriver	Points to a null-terminated string that specifies the MS-DOS filename (without extension) of the device driver (for example, Epson).
lpszDevice	Points to a null-terminated string that specifies the name of the specific device to be supported (for example, Epson FX-80). This parameter is used if the module supports more than one device.
lpszOutput	Points to a null-terminated string that specifies the MS-DOS filename or device name for the physical output medium (file or output port).
lpvInitData	Points to a <u>DEVMODE</u> structure that contains device-specific initialization information for the device driver. The <u>ExtDeviceMode</u> function retrieves this structure already filled in for a given device. The <u>lpvInitData</u> parameter must be NULL if the device driver is to use the default initialization (if any) specified by the user through Windows Control Panel.

### Returns

The return value is the handle of the device context for the specified device if the function is successful. Otherwise, it is NULL.

### Comments

The PRINT.H header file is required if the DEVMODE structure is used.

Device contexts created by using the CreateDC function must be deleted by using the DeleteDC function. All objects selected into the device context after it was created should be selected out and replaced with the original objects before the device context is deleted.

MS-DOS device names follow MS-DOS conventions; an ending colon (:) is recommended, but optional. Windows strips the terminating colon so that a device name ending with a colon is mapped to the same port as the same name without a colon. The driver and port names must not contain leading or trailing spaces.

### Example

The following example uses the CreateDC function to create a device context for a printer, using information returned by the PrintDlg function in a PRINTDLG structure:

```
PRINTDLG pd;
HDC hdc;
LPDEVNAMES lpDevNames;
LPSTR lpszDriverName;
LPSTR lpszDeviceName;
LPSTR lpszPortName;

/*
 * PrintDlg displays the common dialog box for printing. The
 * PRINTDLG structure should be initialized with appropriate values.
 */

PrintDlg(&pd);
lpDevNames = (LPDEVNAMES) GlobalLock(pd.hDevNames);
```

```
lpszDriverName = (LPSTR) lpDevNames + lpDevNames->wDriverOffset;  
lpszDeviceName = (LPSTR) lpDevNames + lpDevNames->wDeviceOffset;  
lpszPortName   = (LPSTR) lpDevNames + lpDevNames->wOutputOffset;  
GlobalUnlock(pd.hDevNames);  
hdc = CreateDC(lpszDriverName, lpszDeviceName, lpszPortName, NULL);
```

See Also

CreateIC, DeleteDC, ExtDeviceMode, PrintDlg, DEVMODE, PRINTDLG



### CreateDIBitmap (3.0)

```
HBITMAP CreateDIBitmap(hdc, lpbmih, dwInit, lpvBits, lpbmi, fnColorUse)
HDC hdc; /* handle of device context */
BITMAPINFOHEADER FAR* lpbmih;
/* address of structure with header
/
DWORD dwInit;
* CBM_INIT to initialize bitmap
/
const void FAR* lpvBits;
* address of array with bitmap values
/
BITMAPINFO FAR* lpbmi;
* address of structure with bitmap data
/
UINT fnColorUse;
* RGB or palette indices
/
```

The CreateDIBitmap function creates a device-specific memory bitmap from a device-independent bitmap (DIB) specification and optionally sets bits in the bitmap.

Parameter	Description						
hdc	Identifies the device context.						
lpbmih	Points to a <u>BITMAPINFOHEADER</u> structure that describes the size and format of the device-independent bitmap.						
dwInit	Specifies whether the memory bitmap is initialized. If this value is CBM_INIT, the function initializes the bitmap with the bits specified by the lpvBits and lpbmi parameters.						
lpvBits	Points to a byte array that contains the initial bitmap values. The format of the bitmap values depends on the biBitCount member of the <u>BITMAPINFOHEADER</u> structure identified by the lpbmi parameter.						
lpbmi	Points to a <u>BITMAPINFO</u> structure that describes the dimensions and color format of the lpvBits parameter. The BITMAPINFO structure contains a <u>BITMAPINFOHEADER</u> structure and an array of <u>RGBQUAD</u> structures specifying the colors in the bitmap.						
fnColorUse	Specifies whether the bmiColors member of the <u>BITMAPINFO</u> structure contains explicit red, green, blue (RGB) values or indices into the currently realized logical palette. The fnColorUse parameter must be one of the following values:						
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>DIB_PAL_COLORS</td><td>The color table consists of an array of 16-bit indices into the currently realized logical palette.</td></tr><tr><td>DIB_RGB_COLORS</td><td>The color table contains literal <u>RGB</u> values.</td></tr></tbody></table>	Value	Meaning	DIB_PAL_COLORS	The color table consists of an array of 16-bit indices into the currently realized logical palette.	DIB_RGB_COLORS	The color table contains literal <u>RGB</u> values.
Value	Meaning						
DIB_PAL_COLORS	The color table consists of an array of 16-bit indices into the currently realized logical palette.						
DIB_RGB_COLORS	The color table contains literal <u>RGB</u> values.						

#### Returns

The return value is the handle of the bitmap if the function is successful. Otherwise, it is NULL.

When it has finished using a bitmap created by CreateDIBitmap, an application should select the bitmap out of the device context and then remove the bitmap by using the DeleteObject function.

#### Example

The following example initializes an array of bits and an array of RGBQUAD structures, allocates memory for the bitmap header and color table, fills in the required members of a BITMAPINFOHEADER structure, and calls the CreateDIBitmap function to create a handle of the bitmap:

```
HANDLE hloc;
PBITMAPINFO pbmi;
HBITMAP hbm;

BYTE aBits[] = { 0x00, 0x00, 0x00, 0x00, /* bottom row */
                 0x01, 0x12, 0x22, 0x11,
                 0x01, 0x12, 0x22, 0x11,
                 0x02, 0x20, 0x00, 0x22,
```

```

    0x02, 0x20, 0x20, 0x22,
    0x02, 0x20, 0x00, 0x22,
    0x01, 0x12, 0x22, 0x11,
    0x01, 0x12, 0x22, 0x11 }; /* top row */

RGBQUAD argbq[] = {{ 255, 0, 0, 0 }, /* blue */
    { 0, 255, 0, 0 }, /* green */
    { 0, 0, 255, 0 }}; /* red */

hloc = LocalAlloc(LMEM_ZEROINIT | LMEM_MOVEABLE,
    sizeof(BITMAPINFOHEADER) + (sizeof(RGBQUAD) * 16));
pbmi = (PBITMAPINFO) LocalLock(hloc);

pbmi->bmiHeader.biSize = sizeof(BITMAPINFOHEADER);
pbmi->bmiHeader.biWidth = 8;
pbmi->bmiHeader.biHeight = 8;
pbmi->bmiHeader.biPlanes = 1;
pbmi->bmiHeader.biBitCount = 4;
pbmi->bmiHeader.biCompression = BI_RGB;

memcpy(pbmi->bmiColors, argbq, sizeof(RGBQUAD) * 3);

hbm = CreateDIBitmap(hdcLocal, (BITMAPINFOHEADER FAR*) pbmi, CBM_INIT,
    aBits, pbmi, DIB_RGB_COLORS);
LocalFree(hloc);
    .
    . /* Use the bitmap handle. */
    .
DeleteObject(hbm);

See Also
CreateBitmap, CreateBitmapIndirect, CreateCompatibleBitmap, CreateDiscardableBitmap, DeleteObject,
BITMAPCOREHEADER, BITMAPCOREINFO, BITMAPINFO, BITMAPINFOHEADER, RGBQUAD

```

DIB\_PAL\_COLORS 1

The color table consists of an array of 16-bit indices into the currently realized logical palette.

DIB\_PAL\_COLORS 1

DIB\_RGB\_COLORS 0

The color table contains literal RGB values.

DIB\_RGB\_COLORS 0

## CreateDIBPatternBrush (3.0)

```
HBRUSH CreateDIBPatternBrush(hglbDIBPacked, fnColorSpec)
HGLOBAL hglbDIBPacked; /* handle of device-independent bitmap */
UINT fnColorSpec;
/* type of color table
/
```

The CreateDIBPatternBrush function creates a brush that has the pattern specified by a device-independent bitmap (DIB). The brush can subsequently be selected for any device that supports raster operations.

Parameter	Description						
hglbDIBPacked	Identifies a global memory object containing a packed device-independent bitmap. A packed DIB consists of a <u>BITMAPINFO</u> structure immediately followed by the array of bytes that define the <u>pixels of the bitmap</u> .						
fnColorSpec	Specifies whether the bmiColors member(s) of the <u>BITMAPINFO</u> structure contain explicit red, green, blue (RGB) values or <u>indices into the currently realized logical palette</u> . This parameter must be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>DIB_PAL_COLORS</td><td>The color table consists of an array of 16-bit indices into the currently realized logical palette.</td></tr><tr><td>DIB_RGB_COLORS</td><td>The color table contains literal <u>RGB</u> values.</td></tr></tbody></table>	Value	Meaning	DIB_PAL_COLORS	The color table consists of an array of 16-bit indices into the currently realized logical palette.	DIB_RGB_COLORS	The color table contains literal <u>RGB</u> values.
Value	Meaning						
DIB_PAL_COLORS	The color table consists of an array of 16-bit indices into the currently realized logical palette.						
DIB_RGB_COLORS	The color table contains literal <u>RGB</u> values.						

### Returns

The return value is the handle of the brush if the function is successful. Otherwise, it is NULL.

### Comments

To retrieve the handle identified by the hglbDIBPacked parameter, an application calls the GlobalAlloc function to allocate a global memory object and then fills the memory with the packed DIB.

Bitmaps used as fill patterns should be 8 pixels by 8 pixels. If such a bitmap is larger, Windows creates a fill pattern using only the bits corresponding to the first 8 rows and 8 columns of pixels in the upper-left corner of the bitmap.

When an application selects a two-color DIB pattern brush into a monochrome device context, Windows ignores the colors specified in the DIB and instead displays the pattern brush, using the current text and background colors of the device context. Pixels mapped to the first color (at offset 0 in the DIB color table) of the DIB are displayed using the text color, and pixels mapped to the second color (at offset 1 in the color table) are displayed using the background color.

When it has finished using a brush created by CreateDIBPatternBrush, an application should remove the brush by using the DeleteObject function.

### Example

The following example retrieves a bitmap named DIBit from the application's resource file, uses the bitmap to create a pattern brush in a call to the CreateDIBPatternBrush function, selects the brush into a device context, and fills a rectangle by using the new brush:

```
HRSRC hrsrc;
HGLOBAL hglbl;
HBRUSH hbr, hbrOld;

hrsrc = FindResource(hinst, "DIBit", RT_BITMAP);
hglbl = LoadResource(hinst, hrsrc);
LockResource(hglbl);

hbr = CreateDIBPatternBrush(hglbl, DIB_RGB_COLORS);
hbrOld = SelectObject(hdc, hbr);
Rectangle(hdc, 10, 10, 100, 100);
UnlockResource(hglbl);
```

### See Also

CreatePatternBrush, DeleteObject, FindResource, GetDeviceCaps, GlobalAlloc, LoadResource, LockResource, SelectObject, SetBkColor, SetTextColor, UnlockResource, BITMAPINFO

## CreateDiscardableBitmap (2.x)

```
HBITMAP CreateDiscardableBitmap(hdc, nWidth, nHeight)
HDC hdc;          /* handle of device context */
int nWidth;       /* bitmap width */
int nHeight;      /*
bitmap height    */
/
```

The CreateDiscardableBitmap function creates a discardable bitmap that is compatible with the given device. The bitmap has the same number of color planes or the same bits-per-pixel format as the device. An application can select this bitmap as the current bitmap for a memory device that is compatible with the one identified by the hdc parameter.

Parameter	Description
hdc	Identifies the device context.
nWidth	Specifies the width, in bits, of the bitmap.
nHeight	Specifies the height, in bits, of the bitmap.

### Returns

The return value is the handle of the bitmap if the function is successful. Otherwise, it is NULL.

### Comments

Windows can discard a bitmap created by this function only if an application has not selected it into a device context. If Windows discards the bitmap when it is not selected and the application later attempts to select it, the SelectObject function will return zero.

Applications should use the DeleteObject function to delete the handle returned by the CreateDiscardableBitmap function, even if Windows has discarded the bitmap.

### See Also

CreateBitmap, CreateBitmapIndirect, CreateDIBitmap, DeleteObject



## CreateEllipticRgn (2.x)

```
HRGN CreateEllipticRgn(nLeftRect, nTopRect, nRightRect, nBottomRect)
int nLeftRect;          /* x-coordinate upper-left corner bounding rectangle */
int nTopRect;           /* y-coordinate upper-left corner bounding rectangle */
int nRightRect;         /* x-coordinate lower-right corner bounding rectangle */
int nBottomRect;       /* y-coordinate lower-right corner bounding rectangle */
```

The CreateEllipticRgn function creates an elliptical region.

Parameter	Description
nLeftRect	Specifies the logical x-coordinate of the upper-left corner of the bounding rectangle of the ellipse.
nTopRect	Specifies the logical y-coordinate of the upper-left corner of the bounding rectangle of the ellipse.
nRightRect	Specifies the logical x-coordinate of the lower-right corner of the bounding rectangle of the ellipse.
nBottomRect	Specifies the logical y-coordinate of the lower-right corner of the bounding rectangle of the ellipse.

### Returns

The return value is the handle of the region if the function is successful. Otherwise, it is NULL.

### Comments

The size of a region is limited to 32,767 by 32,767 logical units or 64K of memory, whichever is smaller.

When it has finished using a region created by using the CreateEllipticRgn function, an application should remove it by using the DeleteObject function.

### See Also

CreateEllipticRgnIndirect, DeleteObject, PaintRgn

## CreateEllipticRgnIndirect (2.x)

```
HRGN CreateEllipticRgnIndirect(lprc)
const RECT FAR* lprc;          /* address of structure with bounding rectangle */
```

The CreateEllipticRgnIndirect function creates an elliptical region.

Parameter	Description
lprc	Points to a <u>RECT</u> structure that contains the logical coordinates of the upper-left and lower-right corners of the bounding rectangle of the ellipse.

### Returns

The return value is the handle of the region if the function is successful. Otherwise, it is NULL.

### Comments

The size of a region is limited to 32,767 by 32,767 logical units or 64K of memory, whichever is smaller.

When it has finished using a region created by CreateEllipticRgnIndirect, an application should remove the region by using the DeleteObject function.

### Example

The following example assigns values to the members of a RECT structure, uses the CreateEllipticRgnIndirect function to create an elliptical region, selects the region into a device context, and then uses the PaintRgn function to display the region:

```
HDC hdc;
RECT rc;
HRGN hrgn;

SetRect(&rc, 10, 10, 200, 50);

hrgn = CreateEllipticRgnIndirect(&rc);
SelectObject(hdc, hrgn);
PaintRgn(hdc, hrgn);
```

See Also  
CreateEllipticRgn, DeleteObject, PaintRgn, RECT

## CreateFont (2.x)

HFONT CreateFont(nHeight, nWidth, nEscapement, nOrientation, fnWeight, fbItalic, fbUnderline, fbStrikeOut, fbCharSet, fbOutputPrecision, fbClipPrecision, fbQuality, fbPitchAndFamily, lpszFace)

```
int nHeight;          /* font height */
int nWidth;           /* character width */
int nEscapement;     /* escapement of line of text */
int nOrientation;   /* angle of base line and x-axis */
int fnWeight;       /* font weight */
BYTE fbItalic;      /* flag for italic attribute */
BYTE fbUnderline;   /* flag for underline attribute */
BYTE fbStrikeOut;   /* flag for strikeout attribute */
BYTE fbCharSet;     /* character set */
BYTE fbOutputPrecision; /* output precision */
BYTE fbClipPrecision; /* clipping precision */
BYTE fbQuality;     /* output quality */
BYTE fbPitchAndFamily; /* pitch and family */
LPCSTR lpszFace;    /* address of typeface name */
```

The CreateFont function creates a logical font that has the specified characteristics. The logical font can subsequently be selected as the font for any device.

Parameter	Description
nHeight	Specifies the requested height, in logical units, for the font. If this parameter is greater than zero, it specifies the cell height of the font. If it is less than zero, it specifies the character height of the font. (Character height is the cell height minus the internal leading. Applications that specify font height in points typically use a negative number for this member.) If this parameter is zero, the font mapper uses a default height. The font mapper chooses the largest physical font that does not exceed the requested size (or the smallest font, if all the fonts exceed the requested size). The absolute value of the nHeight parameter must not exceed 16,384 after it is converted to device units.
nWidth	Specifies the average width, in logical units, of characters in the font. If this parameter is zero, the font mapper chooses a "closest match" default width for the specified font height. (The default width is chosen by matching the aspect

ratio of the device against the digitization aspect ratio of the available fonts. The closest match is determined by the absolute value of the difference.)

nEscapement Specifies the angle, in tenths of degrees, between the escapement vector and the x-axis of the screen surface. The escapement vector is the line through the origins of the first and last characters on a line. The angle is measured counterclockwise from the x-axis.

nOrientation Specifies the angle, in tenths of degrees, between the base line of a character and the x-axis. The angle is measured in a counterclockwise direction from the x-axis for left-handed coordinate systems (that is, MM\_TEXT, in which the y-direction is down) and in a clockwise direction from the x-axis for right-handed coordinate systems (in which the y-direction is up).

fnWeight Specifies the font weight. This parameter can be one of the following values:

Constant	Value
FW_DONTCARE	0
FW_THIN	100
FW_EXTRALIGHT	200
FW_ULTRALIGHT	200
FW_LIGHT	300
FW_NORMAL	400
FW_REGULAR	400
FW_MEDIUM	500
FW_SEMIBOLD	600
FW_DEMIBOLD	600
FW_BOLD	700
FW_EXTRABOLD	800
FW_ULTRABOLD	800
FW_BLACK	900
FW_HEAVY	900

The appearance of the font depends on the typeface. Some fonts have only FW\_NORMAL, FW\_REGULAR, and FW\_BOLD weights. If FW\_DONTCARE is specified, a default weight is used.

fbItalic Specifies an italic font if set to nonzero.

fbUnderline Specifies an underlined font if set to nonzero.

fbStrikeOut Specifies a strikeout font if set to nonzero.

fbCharSet Specifies the character set of the font. The following values are predefined:

Constant	Value
ANSI_CHARSET	0
DEFAULT_CHARSET	1
SYMBOL_CHARSET	2
SHIFTJIS_CHARSET	128
OEM_CHARSET	255

The DEFAULT\_CHARSET value is not used by the font mapper. An application can use this value to allow the name and size of a font to fully describe the logical font. If the specified font name does not exist, a font from any character set can be substituted for the specified font; to avoid unexpected results, applications should use the DEFAULT\_CHARSET value sparingly.

The OEM character set is system-dependent.

Fonts with other character sets may exist in the system. If an application uses a font with an unknown character set, it should not attempt to translate or interpret strings that are to be rendered with that font.

fbOutputPrecision Specifies the requested output precision. The output precision defines how closely the output must match the requested font's height, width, character orientation, escapement, and pitch. This parameter can be one of the following values:

OUT\_CHARACTER\_PRECIS

OUT\_DEFAULT\_PRECIS  
OUT\_DEVICE\_PRECIS  
OUT\_RASTER\_PRECIS  
OUT\_STRING\_PRECIS  
OUT\_STROKE\_PRECIS  
OUT\_TT\_PRECIS

Applications can use the OUT\_DEVICE\_PRECIS, OUT\_RASTER\_PRECIS, and OUT\_TT\_PRECIS values to control how the font mapper chooses a font when the system contains more than one font with a given name. For example, if a system contained a font named Symbol in raster and TrueType form, specifying OUT\_TT\_PRECIS would force the font mapper to choose the TrueType version. (Specifying OUT\_TT\_PRECIS forces the font mapper to choose a TrueType font whenever the specified font name matches a device or raster font, even when there is no TrueType font of the same name.)

fbClipPrecision

Specifies the requested clipping precision. The clipping precision defines how to clip characters that are partially outside the clipping region. This parameter can be one of the following values:

CLIP\_CHARACTER\_PRECIS  
CLIP\_DEFAULT\_PRECIS  
CLIP\_ENCAPSULATE  
CLIP\_LH\_ANGLES  
CLIP\_MASK  
CLIP\_STROKE\_PRECIS  
CLIP\_TT\_ALWAYS

To use an embedded read-only font, applications must specify CLIP\_ENCAPSULATE.

To achieve consistent rotation of device, TrueType, and vector fonts, an application can use the OR operator to combine the CLIP\_LH\_ANGLES value with any of the other fbClipPrecision values. If the CLIP\_LH\_ANGLES bit is set, the rotation for all fonts is dependent on whether the orientation of the coordinate system is left-handed or right-handed. If CLIP\_LH\_ANGLES is not set, device fonts always rotate counterclockwise, but the rotation of other fonts is dependent on the orientation of the coordinate system. (For more information about the orientation of coordinate systems, see the description of the nOrientation parameter.)

fbQuality

Specifies the output quality of the font, which defines how carefully the graphics device interface (GDI) must attempt to match the attributes of a logical font to those of a physical font. This parameter can be one of the following values:

Value	Meaning
DEFAULT_QUALITY	Appearance of the font does not matter.
DRAFT_QUALITY	Appearance of the font is less important than when the PROOF_QUALITY value is used. For GDI raster fonts, scaling is enabled. Bold, italic, underline, and strikeout fonts are synthesized if necessary.
PROOF_QUALITY	Character quality of the font is more important than exact matching of the logical-font attributes. For GDI raster fonts, scaling is disabled and the font closest in size is chosen. Bold, italic, underline, and strikeout fonts are synthesized if necessary.

fbPitchAndFamily

Specifies the pitch and family of the font. The two low-order bits specify the pitch of the font and can be one of the following values:

DEFAULT\_PITCH  
FIXED\_PITCH  
VARIABLE\_PITCH

Applications can set bit 2 (0x04) of the lfPitchAndFamily member to choose a TrueType font.

The four high-order bits specify the font family and can be one of the following values:

Value	Meaning
FF_DECORATIVE	Novelty fonts. Old English is an example.
FF_DONTCARE	Don't care or don't know.
FF_MODERN	Fonts with constant stroke width, with or without serifs. Pica, Elite, and Courier New are examples.
FF_ROMAN	Fonts with variable stroke width and with serifs. Times New Roman and New Century Schoolbook are examples.
FF_SCRIPT	Fonts designed to look like handwriting. Script and Cursive are examples.
FF_SWISS	Fonts with variable stroke width and without serifs. MS Sans Serif is an example.

An application can specify a value for the `fbPitchAndFamily` parameter by using the Boolean OR operator to join a pitch constant with a family constant.

Font families describe the look of a font in a general way. They are intended for specifying fonts when the exact typeface requested is not available.

`lpszFace` Points to a null-terminated string that specifies the typeface name of the font. The length of this string must not exceed `LF_FACESIZE - 1`. The [EnumFontFamilies](#) function can be used to enumerate the typeface names of all currently available fonts. If this parameter is NULL, [GDI](#) uses a device-dependent typeface.

#### Returns

The return value is the handle of the logical font if the function is successful. Otherwise, it is NULL.

#### Comments

The `CreateFont` function creates the handle of a logical font. The font mapper uses this logical font to find the closest match from the fonts available in [GDI](#)'s pool of physical fonts.

Applications can use the default settings for most of these parameters when creating a logical font. The parameters that should always be given specific values are `nHeight` and `lpszFace`. If `nHeight` and `lpszFace` are not set by the application, the logical font that is created is device-dependent.

Fonts created by using the `CreateFont` function must be selected out of any device context in which they were used and then removed by using the [DeleteObject](#) function.

#### Example

The following example sets the mapping mode to `MM_TWIPS` and then uses the `CreateFont` function to create an 18-point logical font:

```
HFONT hfont, hfontOld;
int MapModePrevious, iPtSize = 18;
PSTR pszFace = "MS Serif";

MapModePrevious = SetMapMode(hdc, MM_TWIPS);
hfont = CreateFont(-iPtSize * 20, 0, 0, 0, 0, 0, /* specify pt size */
    0, 0, 0, 0, 0, 0, 0, 0, pszFace); /* and face name only */

hfontOld = SelectObject(hdc, hfont);

TextOut(hdc, 100, -500, pszFace, strlen(pszFace));
SetMapMode(hdc, MapModePrevious);
SelectObject(hdc, hfontOld);
DeleteObject(hfont);
```

#### See Also

[CreateFontIndirect](#), [DeleteObject](#), [EnumFontFamilies](#)

## CreateFontIndirect (2.x)

```
HFONT CreateFontIndirect(lplf)
const LOGFONT FAR* lplf;          /* address of struct. with font attributes */
```

The CreateFontIndirect function creates a logical font that has the characteristics given in the specified structure. The font can subsequently be selected as the current font for any device.

Parameter	Description
lplf	Points to a <u>LOGFONT</u> structure that defines the characteristics of the logical font.

### Returns

The return value is the handle of the logical font if the function is successful. Otherwise, it is NULL.

### Comments

The CreateFontIndirect function creates a logical font that has the characteristics specified in the LOGFONT structure. When the font is selected by using the SelectObject function, the graphics device interface (GDI) font mapper attempts to match the logical font with an existing physical font. If it cannot find an exact match for the logical font, the font mapper provides an alternative whose characteristics match as many of the requested characteristics as possible.

Fonts created by using the CreateFontIndirect function must be selected out of any device context in which they were used and then removed by using the DeleteObject function.

### Example

The following example uses the CreateFontIndirect function to retrieve the handle of a logical font. The nPtSize and pszFace parameters are passed to the function containing this code. The MulDiv and GetDeviceCaps functions are used to convert the specified point size into the correct point size for the MM\_TEXT mapping mode on the current device.

```
HFONT hfont, hfontOld;

PLOGFONT plf = (PLOGFONT) LocalAlloc(LPTR, sizeof(LOGFONT));

plf->lfHeight = -MulDiv(nPtSize, GetDeviceCaps(hdc, LOGPIXELSY), 72);
strcpy(plf->lfFaceName, pszFace);

hfont = CreateFontIndirect(plf);

hfontOld = SelectObject(hdc, hfont);

TextOut(hdc, 10, 50, pszFace, strlen(pszFace));

LocalFree((HLOCAL) plf);
SelectObject(hdc, hfontOld);
DeleteObject(hfont);
```

### See Also

CreateFont, DeleteObject

## CreateHatchBrush (2.x)

```
HBRUSH CreateHatchBrush(fnStyle, clrref)
int fnStyle; /* hatch style of brush */
COLORREF clrref; /* color of brush */
```

The CreateHatchBrush function creates a brush that has the specified hatched pattern and color. The brush can subsequently be selected as the current brush for any device.

Parameter Description  
fnStyle Specifies the hatch style of the brush. This parameter can be one of the following values:

Value	Meaning
<b>HS_BDIAGONAL</b>	45-degree upward hatch (left to right)
<b>HS_CROSS</b>	Horizontal and vertical crosshatch
<b>HS_DIAGCROSS</b>	45-degree crosshatch
<b>HS_FDIAGONAL</b>	45-degree downward hatch (left to right)
<b>HS_HORIZONTAL</b>	Horizontal hatch
<b>HS_VERTICAL</b>	Vertical hatch

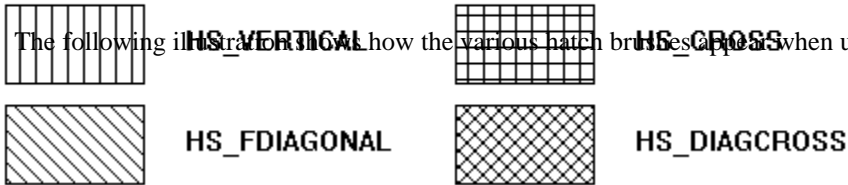
clrref Specifies the foreground color of the brush (the color of the hatches).

### Returns

The return value is the handle of the brush if the function is successful. Otherwise, it is NULL.

<b>Comments</b>
When an application has finished using the brush created by the CreateHatchBrush function, it should select the brush out of the device context and then delete it by using the <u>DeleteObject</u> function.

The following illustration shows how the various hatch brushes appear when used to fill a rectangle:



### Example

The following example creates a hatched brush with green diagonal hatch marks and uses that brush to fill a rectangle:

```
HBRUSH hbr, hbrOld;

hbr = CreateHatchBrush(HS_FDIAGONAL, RGB(0, 255, 0));
hbrOld = SelectObject(hdc, hbr);
Rectangle(hdc, 0, 0, 100, 100);
```

### See Also

CreateBrushIndirect, CreateDIBPatternBrush, CreatePatternBrush, CreateSolidBrush, DeleteObject, SelectObject, RGB



HS\_BDIAGONAL 3

45-degree upward hatch (left to right)

HS\_BDIAGONAL 3

HS\_CROSS 4

Horizontal and vertical crosshatch

HS\_CROSS 4

HS\_DIAGCROSS 5  
45-degree crosshatch

HS\_DIAGCROSS 5

HS\_FDIAGONAL 2

45-degree downward hatch (left to right)

HS\_FDIAGONAL 2



HS\_HORIZONTAL 0

Horizontal hatch

HS\_HORIZONTAL 0

HS\_VERTICAL 1

Vertical hatch

HS\_VERTICAL 1

## CreateIC (2.x)

```
HDC CreateIC(lpszDriver, lpszDevice, lpszOutput, lpvInitData)
LPCSTR lpszDriver;          /* address of driver name          */
LPCSTR lpszDevice;         /* address of device name         */
* address of device name
/
LPCSTR lpszOutput;
* address of filename or port name
/
const void FAR* lpvInitData;
* address of initialization data
/
```

The CreateIC function creates an information context for the specified device. The information context provides a fast way to get information about the device without creating a device context.

Parameter	Description
lpszDriver	Points to a null-terminated string that specifies the MS-DOS filename (without extension) of the device driver (for example, EPSON).
lpszDevice	Points to a null-terminated string that specifies the name of the specific device to be supported (for example, EPSON FX-80). This parameter is used if the module supports more than one device.
lpszOutput	Points to a null-terminated string that specifies the MS-DOS filename or device name for the physical output medium (file or port).
lpvInitData	Points to a <u>DEVMODE</u> structure that contains, initially, device-specific information necessary to initialize the device driver. The <u>ExtDeviceMode</u> function retrieves this structure filled in for a given device. The lpvInitData parameter must be NULL if the device driver is to use the default initialization information (if any) specified by the user through Windows Control Panel.

### Returns

The return value is the handle of an information context for the given device if the function is successful. Otherwise, it is NULL.

### Comments

The PRINT.H header file is required if the DEVMODE structure is used.

MS-DOS device names follow MS-DOS conventions; an ending colon (:) is recommended, but optional. Windows strips the terminating colon so that a device name ending with a colon is mapped to the same port as would be the same name without a colon.

The driver and port names must not contain leading or trailing spaces.

GDI output functions cannot be used with information contexts.

When it has finished using an information context created by CreateIC, an application should remove the information context by using the DeleteDC function.

### Example

The following example uses the CreateIC function to create an information context for the display and then uses the GetDCOrg function to retrieve the origin for the information context:

```
HDC   hdcIC;
DWORD dwOrigin;

hdcIC = CreateIC("DISPLAY", NULL, NULL, NULL);
dwOrigin = GetDCOrg(hdcIC);

DeleteDC(hdcIC);
```

### See Also

CreateDC, DeleteDC, ExtDeviceMode, DEVMODE

## CreateMetaFile (2.x)

```
HDC CreateMetaFile(lpszFile)
LPCSTR lpszFile;      /* address of metafile name */
```

The CreateMetaFile function creates a metafile device context.

Parameter	Description
lpszFile	Points to a null-terminated string that specifies the MS-DOS filename of the metafile to create. If this parameter is NULL, a device context for a memory metafile is returned.

### Returns

The return value is the handle of the metafile device context if the function is successful. Otherwise, it is NULL.

### Comments

When it has finished using a metafile device context created by CreateMetaFile, an application should close it by using the [CloseMetaFile](#) function.

### Example

The following example uses the CreateMetaFile function to create the handle of a device context for a memory metafile, draws a line in that device context, retrieves a handle of the metafile by calling the [CloseMetaFile](#) function, plays the metafile by using the [PlayMetaFile](#) function, and finally deletes the metafile by using the [DeleteMetaFile](#) function:

```
HDC hdcMeta;
HMETAFILE hmf;

hdcMeta = CreateMetaFile(NULL);
MoveTo(hdcMeta, 10, 10);
LineTo(hdcMeta, 100, 100);
hmf = CloseMetaFile(hdcMeta);
PlayMetaFile(hdc, hmf);
DeleteMetaFile(hmf);
```

### See Also

[DeleteMetaFile](#)

## CreatePalette (3.0)

```
HPALETTE CreatePalette(lpplgpl)
const LOGPALETTE FAR* lpplgpl;          /* address of LOGPALETTE structure */
```

The CreatePalette function creates a logical color palette.

Parameter	Description
lpplgpl	Points to a <u>LOGPALETTE</u> structure that contains information about the colors in the logical palette.

### Returns

The return value is the handle of the logical palette if the function is successful. Otherwise, it is NULL.

### Comments

When it has finished using a palette created by CreatePalette, an application should remove the palette by using the DeleteObject function.

### Example

The following example initializes a LOGPALETTE structure and an array of PALETTEENTRY structures, and then uses the CreatePalette function to retrieve a handle of a logical palette:

```
#define NUMENTRIES 128
HPALETTE hpal;
PALETTEENTRY ape[NUMENTRIES];

plgpl = (LOGPALETTE*) LocalAlloc(LPTR,
    sizeof(LOGPALETTE) + cColors * sizeof(PALETTEENTRY));

plgpl->palNumEntries = cColors;
plgpl->palVersion = 0x300;

for (i = 0, red = 0, green = 127, blue = 127; i < NUMENTRIES;
    i++, red += 1, green += 1, blue += 1) {
    ape[i].peRed =
    plgpl->palPalEntry[i].peRed = LOBYTE(red);
    ape[i].peGreen =
    plgpl->palPalEntry[i].peGreen = LOBYTE(green);
    ape[i].peBlue =
    plgpl->palPalEntry[i].peBlue = LOBYTE(blue);
    ape[i].peFlags =
    plgpl->palPalEntry[i].peFlags = PC_RESERVED;
}
hpal = CreatePalette(plgpl);
LocalFree((HLOCAL) plgpl);
.
. /* Use the palette handle. */
.
DeleteObject(hpal);

See Also
DeleteObject
```

## CreatePatternBrush (2.x)

```
HBRUSH CreatePatternBrush(hbmp)
HBITMAP hbmp; /* handle of bitmap */
```

The CreatePatternBrush function creates a brush whose pattern is specified by a bitmap. The brush can subsequently be selected for any device that supports raster operations.

Parameter	Description
hbmp	Identifies the bitmap.

### Returns

The return value is the handle of the brush if the function is successful. Otherwise, it is NULL.

### Comments

The bitmap identified by the hbmp parameter is typically created by using the [CreateBitmap](#), [CreateBitmapIndirect](#), [CreateCompatibleBitmap](#), or [LoadBitmap](#) function.

Bitmaps used as fill patterns should be 8 pixels by 8 pixels. If the bitmap is larger, Windows will use the bits corresponding to only the first 8 rows and 8 columns of pixels in the upper-left corner of the bitmap.

An application can use the [DeleteObject](#) function to remove a pattern brush. This does not affect the associated bitmap, which means the bitmap can be used to create any number of pattern brushes. In any case, when the brush is no longer needed, the application should remove it by using DeleteObject.

A brush created by using a monochrome bitmap (one color plane, one bit per pixel) is drawn using the current text and background colors. Pixels represented by a bit set to 0 are drawn with the current text color, and pixels represented by a bit set to 1 are drawn with the current background color.

### Example

The following example loads a bitmap named Pattern, uses the bitmap to create a pattern brush in a call to the CreatePatternBrush function, selects the brush into a device context, and fills a rectangle by using the new brush:

```
HBITMAP hbmp;
HBRUSH hbr, hbrOld;

hbmp = LoadBitmap(hinst, "Pattern");
hbr = CreatePatternBrush(hbmp);
hbrOld = SelectObject(hdc, hbr);
Rectangle(hdc, 10, 10, 100, 100);
```

### See Also

[CreateBitmap](#), [CreateBitmapIndirect](#), [CreateCompatibleBitmap](#), [CreateDIBPatternBrush](#), [DeleteObject](#), [GetDeviceCaps](#), [LoadBitmap](#), [SelectObject](#), [SetBkColor](#), [SetTextColor](#)



## CreatePen (2.x)

HPEN CreatePen(fnPenStyle, nWidth, clrref)

int fnPenStyle; /\* style of pen \*/

int nWidth;

/\* width of pen \*/

COLORREF clrref;

\* color of pen

/

The CreatePen function creates a pen having the specified style, width, and color. The pen can subsequently be selected as the current pen for any device.

Parameter	Description																
fnPenStyle	Specifies the pen style. This parameter can be one of the following values:																
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td><u>PS_SOLID</u></td> <td>Creates a solid pen.</td> </tr> <tr> <td><u>PS_DASH</u></td> <td>Creates a dashed pen. (Valid only when the pen width is 1.)</td> </tr> <tr> <td><u>PS_DOT</u></td> <td>Creates a dotted pen. (Valid only when the pen width is 1.)</td> </tr> <tr> <td><u>PS_DASHDOT</u></td> <td>Creates a pen with alternating dashes and dots. (Valid only when the pen width is 1.)</td> </tr> <tr> <td><u>PS_DASHDOTDOT</u></td> <td>Creates a pen with alternating dashes and double dots. (Valid only when the pen width is 1.)</td> </tr> <tr> <td><u>PS_NULL</u></td> <td>Creates a null pen.</td> </tr> <tr> <td><u>PS_INSIDEFRAME</u></td> <td>Creates a pen that draws a line inside the frame of closed shapes produced by graphics device interface (GDI) output functions that specify a bounding rectangle (for example, the <u>Ellipse</u>, <u>Rectangle</u>, <u>RoundRect</u>, <u>Pie</u>, and <u>Chord</u> functions). When this style is used with GDI output functions that do not specify a bounding rectangle (for example, the <u>LineTo</u> function), the drawing area of the pen is not limited by a frame.</td> </tr> </tbody> </table>	Value	Meaning	<u>PS_SOLID</u>	Creates a solid pen.	<u>PS_DASH</u>	Creates a dashed pen. (Valid only when the pen width is 1.)	<u>PS_DOT</u>	Creates a dotted pen. (Valid only when the pen width is 1.)	<u>PS_DASHDOT</u>	Creates a pen with alternating dashes and dots. (Valid only when the pen width is 1.)	<u>PS_DASHDOTDOT</u>	Creates a pen with alternating dashes and double dots. (Valid only when the pen width is 1.)	<u>PS_NULL</u>	Creates a null pen.	<u>PS_INSIDEFRAME</u>	Creates a pen that draws a line inside the frame of closed shapes produced by graphics device interface (GDI) output functions that specify a bounding rectangle (for example, the <u>Ellipse</u> , <u>Rectangle</u> , <u>RoundRect</u> , <u>Pie</u> , and <u>Chord</u> functions). When this style is used with GDI output functions that do not specify a bounding rectangle (for example, the <u>LineTo</u> function), the drawing area of the pen is not limited by a frame.
Value	Meaning																
<u>PS_SOLID</u>	Creates a solid pen.																
<u>PS_DASH</u>	Creates a dashed pen. (Valid only when the pen width is 1.)																
<u>PS_DOT</u>	Creates a dotted pen. (Valid only when the pen width is 1.)																
<u>PS_DASHDOT</u>	Creates a pen with alternating dashes and dots. (Valid only when the pen width is 1.)																
<u>PS_DASHDOTDOT</u>	Creates a pen with alternating dashes and double dots. (Valid only when the pen width is 1.)																
<u>PS_NULL</u>	Creates a null pen.																
<u>PS_INSIDEFRAME</u>	Creates a pen that draws a line inside the frame of closed shapes produced by graphics device interface (GDI) output functions that specify a bounding rectangle (for example, the <u>Ellipse</u> , <u>Rectangle</u> , <u>RoundRect</u> , <u>Pie</u> , and <u>Chord</u> functions). When this style is used with GDI output functions that do not specify a bounding rectangle (for example, the <u>LineTo</u> function), the drawing area of the pen is not limited by a frame.																
nWidth	Specifies the width, in logical units, of the pen. If this value is zero, the width in device units is always one pixel, regardless of the mapping mode.																
clrref	Specifies the color of the pen.																

### Returns

The return value is the handle of the pen if the function is successful. Otherwise, it is NULL.

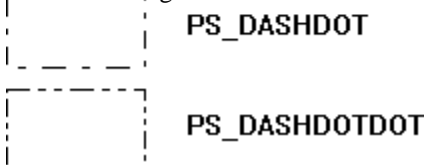
### Comments

Pens whose width is greater than one pixel always have the PS\_NULL, PS\_SOLID, or PS\_INSIDEFRAME style.

If a pen has the PS\_INSIDEFRAME style and a color that does not match a color in the logical color table, the pen is drawn with a dithered color. The PS\_SOLID pen style cannot be used to create a pen with a dithered color. The style PS\_INSIDEFRAME is identical to PS\_SOLID if the pen width is less than or equal to 1.

When it has finished using a pen created by CreatePen, an application should remove the pen by using the DeleteObject function.

The following illustration shows how the various system pens appear when used to draw a rectangle.



### Example

The following example uses the CreatePen function to create a solid blue pen 6 units wide, selects the pen into a device context, and then uses the pen to draw a rectangle:

```
HPEN hpen, hpenOld;
```

```
hpen = CreatePen(PS_SOLID, 6, RGB(0, 0, 255));
```

```
hpenOld = SelectObject(hdc, hpen);
```

```
Rectangle(hdc, 10, 10, 100, 100);
```

```
SelectObject(hdc, hpenOld);
```

```
DeleteObject(hpen);
```

See Also

CreatePenIndirect, DeleteObject, Ellipse, Rectangle, RoundRect, RGB

PS\_SOLID 0

Creates a solid pen.

PS\_SOLID 0

PS\_DASH 1

Creates a dashed pen. (Valid only when the pen width is 1.)

PS\_DASH 1

PS\_DOT 2

Creates a dotted pen. (Valid only when the pen width is 1.)

PS\_DOT 2



PS\_DASHDOT 3

Creates a pen with alternating dashes and dots. (Valid only when the pen width is 1.)

PS\_DASHDOT 3

PS\_DASHDOTDOT 4

Creates a pen with alternating dashes and double dots. (Valid only when the pen width is 1.)

PS\_DASHDOTDOT 4

PS\_NULL 5

Creates a null pen.

PS\_NULL 5

PS\_INSIDEFRAME 6

Creates a pen that draws a line inside the frame of closed shapes produced by graphics device interface (GDI) output functions that specify a bounding rectangle (for example, the Ellipse, Rectangle, RoundRect, Pie, and Chord functions). When this style is used with GDI output functions that do not specify a bounding rectangle (for example, the LineTo function), the drawing area of the pen is not limited by a frame.

PS\_INSIDEFRAME 6



## CreatePenIndirect (2.x)

```
HPEN CreatePenIndirect(lplogpn)
LOGPEN FAR* lplogpn;          /* address of structure with pen data */
```

The CreatePenIndirect function creates a pen that has the style, width, and color given in the specified structure.

Parameter	Description
lplogpn	Points to the <u>LOGPEN</u> structure that contains information about the pen.

### Returns

The return value is the handle of the pen if the function is successful. Otherwise, it is NULL.

### Comments

Pens whose width is greater than 1 pixel always have the PS\_NULL, PS\_SOLID, or PS\_INSIDEFRAME style.

If a pen has the PS\_INSIDEFRAME style and a color that does not match a color in the logical color table, the pen is drawn with a dithered color. The PS\_INSIDEFRAME style is identical to PS\_SOLID if the pen width is less than or equal to 1.

When it has finished using a pen created by CreatePenIndirect, an application should remove the pen by using the DeleteObject function.

### Example

The following example fills a LOGPEN structure with values defining a solid red pen 10 logical units wide, uses the CreatePenIndirect function to create this pen, selects the pen into a device context, and then uses the pen to draw a rectangle:

```
LOGPEN lp;
HPEN hpen, hpenOld;

lp.lopnStyle = PS_SOLID;
lp.lopnWidth.x = 10;
lp.lopnWidth.y = 0;      /* y-dimension not used */
lp.lopnColor = RGB(255, 0, 0);

hpen = CreatePenIndirect(&lp);
hpenOld = SelectObject(hdc, hpen);
Rectangle(hdc, 10, 10, 100, 100);
```

### See Also

CreatePen, DeleteObject, LOGPEN, RGB

## CreatePolygonRgn (2.x)

```
HRGN CreatePolygonRgn(lppt, cPoints, fnPolyFillMode)
const POINT FAR* lppt;          /* address of array of points */
int cPoints;                    /* number of
points in array
/
int fnPolyFillMode;
* polygon-filling mode
/
```

The CreatePolygonRgn function creates a polygonal region. The system closes the polygon automatically, if necessary, by drawing a line from the last vertex to the first.

Parameter	Description
lppt	Points to an array of <u>POINT</u> structures. Each structure specifies the x-coordinate and y-coordinate of one vertex of the polygon.
cPoints	Specifies the number of <u>POINT</u> structures in the array pointed to by the lppt parameter.
fnPolyFillMode	Specifies the polygon-filling mode. This value may be either ALTERNATE or WINDING.

### Returns

The return value is the handle of the region if the function is successful. Otherwise, it is NULL.

### Comments

The size of a region is limited to 32,767 by 32,767 logical units or 64K of memory, whichever is smaller.

When the polygon-filling mode is ALTERNATE, the system fills the area between odd-numbered and even-numbered polygon sides on each scan line. That is, the system fills the area between the first and second side, between the third and fourth side, and so on.

When the polygon-filling mode is WINDING, the system uses the direction in which a figure was drawn to determine whether to fill an area. Each line segment in a polygon is drawn in either a clockwise or a counterclockwise direction. Whenever an imaginary line drawn from an enclosed area to the outside of a figure passes through a clockwise line segment, the system increments a count (increases it by one); when the line passes through a counterclockwise line segment, the system decrements the count. The area is filled if the count is nonzero when the line reaches the outside of the figure.

When it has finished using a region created by CreatePolygonRgn, an application should remove the region by using the DeleteObject function.

### Example

The following example fills an array of POINT structures with the coordinates of a five-pointed star, uses this array in a call to the CreatePolygonRgn function, selects the region into a device context, and then uses the PaintRgn function to display the region:

```
HDC hdc;
HRGN hrgn;
POINT apts[5] = {{ 200, 100 },
  { 300, 200 },
  { 100, 100 },
  { 300, 100 },
  { 100, 200 }};

hrgn = CreatePolygonRgn(apts, /* array of points */
  sizeof(apts) / sizeof(POINT), /* number of points */
  ALTERNATE); /* alternate mode */
SelectObject(hdc, hrgn);
PaintRgn(hdc, hrgn);
```

### See Also

CreatePolyPolygonRgn, DeleteObject, Polygon, SetPolyFillMode, POINT

### CreatePolyPolygonRgn (3.0)

```
HRGN CreatePolyPolygonRgn(lppt, lpnPolyCount, cIntegers, fnPolyFillMode)
const POINT FAR* lppt;          /* address of structure of points */
const int FAR* lpnPolyCount;    /* address of array of vertex data */
/
int cIntegers;
/* number of integers in array */
/
int fnPolyFillMode;
/* polygon-filling mode */
/
```

The CreatePolyPolygonRgn function creates a region consisting of a series of closed polygons. The polygons may be disjoint, or they may overlap.

Parameter	Description
lppt	Points to an array of <b>POINT</b> structures that define the vertices of the polygons. Each polygon must be explicitly closed, because the system does not close them automatically. The polygons are specified consecutively.
lpnPolyCount	Points to an array of integers. The first integer specifies the number of vertices in the first polygon in the array pointed to by the lppt parameter, the second integer specifies the number of vertices in the second polygon, and so on.
cIntegers	Specifies the total number of integers in the array pointed to by the lpnPolyCount parameter.
fnPolyFillMode	Specifies the polygon-filling mode. This value may be either <b>ALTERNATE</b> or <b>WINDING</b> .

#### Returns

The return value is the handle of the region if the function is successful. Otherwise, it is **NULL**.

#### Comments

The size of a region is limited to 32,767 by 32,767 logical units or 64K of memory, whichever is smaller.

When the polygon-filling mode is **ALTERNATE**, the system fills the area between odd-numbered and even-numbered polygon sides on each scan line. That is, the system fills the area between the first and second side, between the third and fourth side, and so on.

When the polygon-filling mode is **WINDING**, the system uses the direction in which a figure was drawn to determine whether to fill an area. Each line segment in a polygon is drawn in either a clockwise or a counterclockwise direction. Whenever an imaginary line drawn from an enclosed area to the outside of a figure passes through a clockwise line segment, the system increments a count (increases it by one); when the line passes through a counterclockwise line segment, the system decrements the count. The area is filled if the count is nonzero when the line reaches the outside of the figure.

When it has finished using a region created by CreatePolyPolygonRgn, an application should remove the region by using the DeleteObject function.

#### Example

The following example fills an array of **POINT** structures with the coordinates of a five-pointed star and a rectangle, uses this array in a call to the CreatePolyPolygonRgn function, selects the region into a device context, and then uses the PaintRgn function to display the region:

```
HDC hdc;
HRGN hrgn;
int aVertices[2] = { 6, 5 };
POINT apts[11] = { { 200, 10 },
    { 300, 200 },
    { 100, 100 }, /* Star figure, manually closed */
    { 300, 100 },
    { 100, 200 },
    { 200, 10 },
    { 10, 150 },
```

```
{ 350, 150 },  
{ 350, 170 }, /* Rectangle, manually closed */  
{ 10, 170 },  
{ 10, 150 } };
```

```
hrgn = CreatePolyPolygonRgn(apts, /* array of points*/  
    aVertices, /* array of vertices */  
    sizeof(aVertices) / sizeof(int), /* integers in vertex array */  
    ALTERNATE); /* alternate mode */
```

```
SelectObject(hdc, hrgn);
```

```
PaintRgn(hdc, hrgn);
```

See Also

CreatePolygonRgn, DeleteObject, PolyPolygon, SetPolyFillMode, POINT

## CreateRectRgn (2.x)

```
HRGN CreateRectRgn(nLeftRect, nTopRect, nRightRect, nBottomRect)
int nLeftRect;          /* x-coordinate upper-left corner of region */
int nTopRect;           /* y-coordinate
upper-left corner of region
/
int nRightRect;
* x-coordinate lower-right corner of region
/
int nBottomRect;
* y-coordinate lower-right corner of region
/
```

The CreateRectRgn function creates a rectangular region.

Parameter	Description
nLeftRect	Specifies the logical x-coordinate of the upper-left corner of the region.
nTopRect	Specifies the logical y-coordinate of the upper-left corner of the region.
nRightRect	Specifies the logical x-coordinate of the lower-right corner of the region.
nBottomRect	Specifies the logical y-coordinate of the lower-right corner of the region.

### Returns

The return value is the handle of a rectangular region if the function is successful. Otherwise, it is NULL.

### Comments

The size of a region is limited to 32,767 by 32,767 logical units or 64K of memory, whichever is smaller.

When it has finished using a region created by CreateRectRgn, an application should remove the region by using the [DeleteObject](#) function.

### Example

The following example uses the CreateRectRgn function to create a rectangular region, selects the region into a device context, and then uses the [PaintRgn](#) function to display the region:

```
HDC hdc;
HRGN hrgn;

hrgn = CreateRectRgn(10, 10, 110, 110);
SelectObject(hdc, hrgn);
PaintRgn(hdc, hrgn);
```

### See Also

[CreateRectRgnIndirect](#), [CreateRoundRectRgn](#), [DeleteObject](#), [PaintRgn](#)

## CreateRectRgnIndirect (2.x)

```
HRGN CreateRectRgnIndirect(lprc)
const RECT FAR* lprc;          /* address of structure with region */
```

The CreateRectRgnIndirect function creates a rectangular region by using a RECT structure.

Parameter	Description
lprc	Points to a <u>RECT</u> structure that contains the logical coordinates of the upper-left and lower-right corners of the region.

### Returns

The return value is the handle of the rectangular region if the function is successful. Otherwise, it is NULL.

### Comments

The size of a region is limited to 32,767 by 32,767 logical units or 64K of memory, whichever is smaller.

When it has finished using a region created by CreateRectRgnIndirect, an application should remove the region by using the DeleteObject function.

### Example

The following example assigns values to the members of a RECT structure, uses the CreateRectRgnIndirect function to create a rectangular region, selects the region into a device context, and then uses the PaintRgn function to display the region:

```
RECT rc;
HRGN hrgn;
```

```
SetRect(&rc, 10, 10, 200, 50);
```

```
hrgn = CreateRectRgnIndirect(&rc);
```

```
SelectObject(hdc, hrgn);
```

```
PaintRgn(hdc, hrgn);
```

### See Also

CreateRectRgn, CreateRoundRectRgn, DeleteObject, PaintRgn, RECT

### CreateRoundRectRgn (3.0)

```
HRGN CreateRoundRectRgn(nLeftRect, nTopRect, nRightRect, nBottomRect, nWidthEllipse,
                        nHeightEllipse)
int nLeftRect;          /* x-coordinate upper-left corner of region */
int nTopRect;           /* y-coordinate
upper-left corner of region
/
int nRightRect;
* x-coordinate lower-right corner of region
/
int nBottomRect;
* y-coordinate lower-right corner of region
/
int nWidthEllipse;
* height of ellipse for rounded corners
/
int nHeightEllipse;
* width of ellipse for rounded corners
/
```

The CreateRoundRectRgn function creates a rectangular region with rounded corners.

Parameter	Description
nLeftRect	Specifies the logical x-coordinate of the upper-left corner of the region.
nTopRect	Specifies the logical y-coordinate of the upper-left corner of the region.
nRightRect	Specifies the logical x-coordinate of the lower-right corner of the region.
nBottomRect	Specifies the logical y-coordinate of the lower-right corner of the region.
nWidthEllipse	Specifies the width of the ellipse used to create the rounded corners.
nHeightEllipse	Specifies the height of the ellipse used to create the rounded corners.

#### Returns

The return value is the handle of the region if the function is successful. Otherwise, it is NULL.

#### Comments

The size of a region is limited to 32,767 by 32,767 logical units or 64K of memory, whichever is smaller.

When it has finished using a region created by CreateRoundRectRgn, an application should remove the region by using the [DeleteObject](#) function.

#### Example

The following example uses the CreateRoundRectRgn function to create a region, selects the region into a device context, and then uses the [PaintRgn](#) function to display the region:

```
HRGN hrgn;
int nEllipWidth = 10;
int nEllipHeight = 30;

hrgn = CreateRoundRectRgn(10, 10, 110, 110,
    nEllipWidth, nEllipHeight);
SelectObject(hdc, hrgn);
PaintRgn(hdc, hrgn);
```

#### See Also

[CreateRectRgn](#), [CreateRectRgnIndirect](#), [DeleteObject](#), [PaintRgn](#)

### CreateScalableFontResource (3.1)

```
BOOL CreateScalableFontResource(fHidden, lpszResourceFile, lpszFontFile, lpszCurrentPath)
UINT fHidden; /* flag for read-only embedded font */
LPCSTR lpszResourceFile;
/* address of filename of font resource
/
LPCSTR lpszFontFile;
/* address of filename of scalable font
/
LPCSTR lpszCurrentPath;
/* address of path to font file
/
```

The CreateScalableFontResource function creates a font resource file for the specified scalable font file.

Parameter	Description						
fHidden	Specifies whether the font is a read-only embedded font. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>The font has read-write permission.</td></tr><tr><td>1</td><td>The font has read-only permission and should be hidden from other applications in the system. When this flag is set, the font is not enumerated by the <u>EnumFonts</u> or <u>EnumFontFamilies</u> function.</td></tr></tbody></table>	Value	Meaning	0	The font has read-write permission.	1	The font has read-only permission and should be hidden from other applications in the system. When this flag is set, the font is not enumerated by the <u>EnumFonts</u> or <u>EnumFontFamilies</u> function.
Value	Meaning						
0	The font has read-write permission.						
1	The font has read-only permission and should be hidden from other applications in the system. When this flag is set, the font is not enumerated by the <u>EnumFonts</u> or <u>EnumFontFamilies</u> function.						
lpszResourceFile	Points to a null-terminated string specifying the name of the font resource file that this function creates.						
lpszFontFile	Points to a null-terminated string specifying the scalable font file this function uses to create the font resource file. This parameter must specify either the filename and extension or a full path and filename, including drive and filename extension.						
lpszCurrentPath	Points to a null-terminated string specifying either the path to the scalable font file specified in the lpszFontFile parameter or NULL, if lpszFontFile specifies a full path.						

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

An application must use the CreateScalableFontResource function to create a font resource file before installing an embedded font. Font resource files for fonts with read-write permission should use the .FOT filename extension. Font resource files for read-only fonts should use a different extension (for example, .FOR) and should be hidden from other applications in the system by specifying 1 for the fHidden parameter. The font resource files can be installed by using the AddFontResource function.

When the lpszFontFile parameter specifies only a filename and extension, the lpszCurrentPath parameter must specify a path. When the lpszFontFile parameter specifies a full path, the lpszCurrentPath parameter must be NULL or a pointer to NULL.

When only a filename and extension is specified in the lpszFontFile parameter and a path is specified in the lpszCurrentPath parameter, the string in lpszFontFile is copied into the .FOT file as the .TTF file that belongs to this resource. When the AddFontResource function is called, the system assumes that the .TTF file has been copied into the SYSTEM directory (or into the main Windows directory in the case of a network installation). The .TTF file need not be in this directory when the CreateScalableFontResource function is called, because the lpszCurrentPath parameter contains the directory information. A resource created in this manner does not contain absolute path information and can be used in any Windows installation.

When a path is specified in the lpszFontFile parameter and NULL is specified in the lpszCurrentPath parameter, the string in lpszFontFile is copied into the .FOT file. In this case, when the AddFontResource function is called, the .TTF file must be at the location specified in the lpszFontFile parameter when the CreateScalableFontResource function was called; the lpszCurrentPath parameter is not needed. A resource created in this manner contains absolute references to paths and drives and will not work if the .TTF file is moved to a different location.

The CreateScalableFontResource function supports only TrueType scalable fonts.



## Example

The following example shows how to create a TrueType font file in the SYSTEM directory of the Windows startup directory:

```
CreateScalableFontResource(0, "c:\\windows\\system\\font.fot",  
    "font.ttr", "c:\\windows\\system");
```

```
AddFontResource("c:\\windows\\system\\font.fot");
```

The following example shows how to create a TrueType font file in a specified directory:

```
CreateScalableFontResource(0, "c:\\windows\\system\\font.fot",  
    "c:\\fontdir\\font.ttr", NULL);
```

```
AddFontResource("c:\\windows\\system\\font.fot");
```

The following example shows how to work with a standard embedded font:

```
HFONT hfont;
```

```
/* Extract .TTF file into C:\\MYDIR\\FONT.TTR. */
```

```
CreateScalableFontResource(0, "font.fot", "c:\\mydir\\font.ttr", NULL)  
;
```

```
AddFontResource("font.fot");
```

```
hfont = CreateFont(..., CLIP_DEFAULT_PRECIS, ..., "FONT");
```

```
    . /* Use the font. */
```

```
    .
```

```
DeleteObject(hfont);
```

```
RemoveFontResource("font.fot");
```

```
    . /* Delete C:\\MYDIR\\FONT.FOT and C:\\MYDIR\\FONT.TTR. */
```

```
    .
```

The following example shows how to work with a read-only embedded font:

```
HFONT hfont;
```

```
/* Extract.TTF file into C:\\MYDIR\\FONT.TTR. */
```

```
CreateScalableFontResource(1, "font.for", "c:\\mydir\\font.ttr", NULL)  
;
```

```
AddFontResource("font.for");
```

```
hfont = CreateFont(..., CLIP_EMBEDDED, ..., "FONT");
```

```
    . /* Use the font. */
```

```
    .
```

```
DeleteObject(hfont);
```

```
RemoveFontResource("font.for");
```

```
    . /* Delete C:\\MYDIR\\FONT.FOR and C:\\MYDIR\\FONT.TTR. */
```

```
    .
```

See Also

AddFontResource

## CreateSolidBrush (2.x)

```
HBRUSH CreateSolidBrush(clrref)
COLORREF clrref;      /* brush color */
```

The CreateSolidBrush function creates a brush that has a specified solid color. The brush can subsequently be selected as the current brush for any device.

Parameter	Description
clrref	Specifies the color of the brush.

### Returns

The return value is the handle of the brush if the function is successful. Otherwise, it is NULL.

### Comments

When an application has finished using the brush created by CreateSolidBrush, it should select the brush out of the device context and then remove it by using the [DeleteObject](#) function.

### Example

The following example uses the CreateSolidBrush function to create a green brush, selects the brush into a device context, and then uses the brush to fill a rectangle:

```
HBRUSH hbrOld;
HBRUSH hbr;

hbr = CreateSolidBrush(RGB(0, 255, 0));
hbrOld = SelectObject(hdc, hbr);
Rectangle(hdc, 10, 10, 100, 100);
```

### See Also

[CreateBrushIndirect](#), [CreateDIBPatternBrush](#), [CreateHatchBrush](#), [CreatePatternBrush](#), [DeleteObject](#), [RGB](#)

## DeleteDC (2.x)

```
BOOL DeleteDC(hdc)
HDC hdc;      /* handle of device context */
```

The DeleteDC function deletes the given device context.

Parameter	Description
hdc	Identifies the device context.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

If the hdc parameter identifies the last device context for a given device, the device is notified and all storage and system resources used by the device are released.

An application must not delete a device context whose handle was retrieved by calling the GetDC function. Instead, the application must call the ReleaseDC function to free the device context.

An application should not call DeleteDC if the application has selected objects into the device context. Objects must be selected out of the device context before it is deleted.

### Example

The following example uses the CreateDC function to create a device context for a printer and then calls the DeleteDC function when the device context is no longer needed:

```
/* Retrieves a device context for a printer. */
hdcPrinter = CreateDC(lpDriverName, lpDeviceName, lpOutput,
    lpInitData);
.
. /* Use the device context. */
.

/* Delete the device context. */

DeleteDC(hdcPrinter);
```

### See Also

CreateDC, GetDC, ReleaseDC

## DeleteMetaFile (2.x)

```
BOOL DeleteMetaFile(hmf)
HMETAFILE hmf;          /* handle of metafile */
```

The DeleteMetaFile function invalidates the given metafile handle.

Parameter	Description
hmf	Identifies the metafile to be deleted.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The DeleteMetaFile function does not destroy a metafile that is saved on disk. After calling DeleteMetaFile, an application can retrieve a new handle of a disk-based metafile by calling the GetMetaFile function.

### Example

The following example uses the CreateMetaFile function to create the handle of a memory metafile device context, draws a line in that device context, retrieves a handle of the metafile by calling the CloseMetaFile function, plays the metafile by using the PlayMetaFile function, and finally deletes the metafile by using DeleteMetaFile:

```
HDC hdcMeta;
HMETAFILE hmf;

hdcMeta = CreateMetaFile(NULL);
MoveTo(hdcMeta, 10, 10);
LineTo(hdcMeta, 100, 100);
hmf = CloseMetaFile(hdcMeta);
PlayMetaFile(hdc, hmf);
DeleteMetaFile(hmf);
```

### See Also

CreateMetaFile, GetMetaFile

## DeleteObject (2.x)

```
BOOL DeleteObject(hgdiobj)
HGDIOBJ hgdiobj;    /* handle of object to delete    */
```

The DeleteObject function deletes an object from memory by freeing all system storage associated with the object. (Objects include pens, brushes, fonts, bitmaps, regions, and palettes.)

Parameter	Description
hgdiobj	Identifies a pen, brush, font, bitmap, region, or palette.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

After the object is deleted, the handle given in the hgdiobj parameter is no longer valid.

An application should not delete an object that is currently selected into a device context.

When a pattern brush is deleted, the bitmap associated with the brush is not deleted. The bitmap must be deleted independently.

### Example

The following example creates a pen, selects it into a device context, and uses the pen to draw a rectangle. To delete the pen, the original pen is selected back into the device context and the DeleteObject function is called.

```
HPEN hpen, hpenOld;
```

```
hpen = CreatePen(PS_SOLID, 6, RGB(0, 0, 255));
hpenOld = SelectObject(hdc, hpen);
```

```
Rectangle(hdc, 10, 10, 100, 100);
```

```
SelectObject(hdc, hpenOld);
DeleteObject(hpen);
```

### See Also

[SelectObject](#), [RGB](#)

## DeviceCapabilities (3.0)

```
#include <print.h>

DWORD DeviceCapabilities(lpszDevice, lpszPort, fwCapability, lpszOutput, lpdm)
LPSTR lpszDevice;          /* address of device-name string */
LPSTR lpszPort;           /*
address of port-name string
/
WORD fwCapability;
* device capability to query
/
LPSTR lpszOutput;
* address of the output
/
LPDEVMODE lpdm;
* address of structure with device data
/
```

The DeviceCapabilities function retrieves the capabilities of the printer device driver.

Parameter	Description																
lpszDevice	Points to a null-terminated string that contains the name of the printer device, such as PCL/HP LaserJet.																
lpszPort	Points to a null-terminated string that contains the name of the port to which the device is connected, such as LPT1.																
fwCapability	Specifies the capabilities to query. This parameter can be one of the following values:																
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>DC_BINNAMES</u></td><td>Copies an array containing a list of the names of the paper bins. This array is in the form char PaperNames[cBinMax][cchBinName] where cchBinName is 24. If the lpszOutput parameter is NULL, the return value is the number of bin entries required. Otherwise, the return value is the number of bins copied.</td></tr><tr><td><u>DC_BINS</u></td><td>Retrieves a list of available bins. The function copies the list to the lpszOutput parameter as a <u>WORD</u> array. If lpszOutput is NULL, the function returns the number of supported bins to allow the application the opportunity to allocate a buffer with the correct size. For more information about these bins, see the description of the dmDefaultSource member of the <u>DEVMODE</u> structure.</td></tr><tr><td><u>DC_COPIES</u></td><td>Returns the number of copies the device can print.</td></tr><tr><td><u>DC_DRIVER</u></td><td>Returns the version number of the printer driver.</td></tr><tr><td><u>DC_DUPLEX</u></td><td>Returns the level of duplex support. The function returns 1 if the printer is capable of duplex printing. Otherwise, the return value is zero.</td></tr><tr><td><u>DC_ENUMRESOLUTIONS</u></td><td>Returns a list of available resolutions. If lpszOutput is NULL, the function returns the number of available resolution configurations. Resolutions are represented by pairs of <u>LONG</u> integers representing the horizontal and vertical resolutions (specified in dots per inch).</td></tr><tr><td><u>DC_EXTRA</u></td><td>Returns the number of bytes required for the device-specific portion of the <u>DEVMODE</u> structure for the printer driver.</td></tr></tbody></table>	Value	Meaning	<u>DC_BINNAMES</u>	Copies an array containing a list of the names of the paper bins. This array is in the form char PaperNames[cBinMax][cchBinName] where cchBinName is 24. If the lpszOutput parameter is NULL, the return value is the number of bin entries required. Otherwise, the return value is the number of bins copied.	<u>DC_BINS</u>	Retrieves a list of available bins. The function copies the list to the lpszOutput parameter as a <u>WORD</u> array. If lpszOutput is NULL, the function returns the number of supported bins to allow the application the opportunity to allocate a buffer with the correct size. For more information about these bins, see the description of the dmDefaultSource member of the <u>DEVMODE</u> structure.	<u>DC_COPIES</u>	Returns the number of copies the device can print.	<u>DC_DRIVER</u>	Returns the version number of the printer driver.	<u>DC_DUPLEX</u>	Returns the level of duplex support. The function returns 1 if the printer is capable of duplex printing. Otherwise, the return value is zero.	<u>DC_ENUMRESOLUTIONS</u>	Returns a list of available resolutions. If lpszOutput is NULL, the function returns the number of available resolution configurations. Resolutions are represented by pairs of <u>LONG</u> integers representing the horizontal and vertical resolutions (specified in dots per inch).	<u>DC_EXTRA</u>	Returns the number of bytes required for the device-specific portion of the <u>DEVMODE</u> structure for the printer driver.
Value	Meaning																
<u>DC_BINNAMES</u>	Copies an array containing a list of the names of the paper bins. This array is in the form char PaperNames[cBinMax][cchBinName] where cchBinName is 24. If the lpszOutput parameter is NULL, the return value is the number of bin entries required. Otherwise, the return value is the number of bins copied.																
<u>DC_BINS</u>	Retrieves a list of available bins. The function copies the list to the lpszOutput parameter as a <u>WORD</u> array. If lpszOutput is NULL, the function returns the number of supported bins to allow the application the opportunity to allocate a buffer with the correct size. For more information about these bins, see the description of the dmDefaultSource member of the <u>DEVMODE</u> structure.																
<u>DC_COPIES</u>	Returns the number of copies the device can print.																
<u>DC_DRIVER</u>	Returns the version number of the printer driver.																
<u>DC_DUPLEX</u>	Returns the level of duplex support. The function returns 1 if the printer is capable of duplex printing. Otherwise, the return value is zero.																
<u>DC_ENUMRESOLUTIONS</u>	Returns a list of available resolutions. If lpszOutput is NULL, the function returns the number of available resolution configurations. Resolutions are represented by pairs of <u>LONG</u> integers representing the horizontal and vertical resolutions (specified in dots per inch).																
<u>DC_EXTRA</u>	Returns the number of bytes required for the device-specific portion of the <u>DEVMODE</u> structure for the printer driver.																

<u>DC_FIELDS</u>	Returns the dmFields member of the printer driver's <u>DEVMODE</u> structure. The dmFields member indicates which fields in the device-independent portion of the structure are supported by the printer driver.								
<u>DC_FILEDEPENDENCIES</u>	Returns a list of files that also need to be loaded when a driver is installed. If the lpszOutput parameter is NULL, the function returns the number of files. Otherwise, lpszOutput points to an array of filenames in the form char [chFileName, 64]. Each filename is a null-terminated string.								
<u>DC_MAXEXTENT</u>	Returns a <u>POINT</u> structure containing the maximum paper size that the dmPaperLength and dmPaperWidth members of the printer driver's <u>DEVMODE</u> structure can specify.								
<u>DC_MINEXTENT</u>	Returns a <u>POINT</u> structure containing the minimum paper size that the dmPaperLength and dmPaperWidth members of the printer driver's <u>DEVMODE</u> structure can specify.								
<u>DC_ORIENTATION</u>	Returns the relationship between portrait and landscape orientations for a device, in terms of the number of degrees that portrait orientation is rotated counterclockwise to produce landscape orientation. The return value can be one of the following: <table border="0" style="margin-left: 20px;"> <thead> <tr> <th style="text-align: left;">Value</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No landscape orientation.</td> </tr> <tr> <td>90</td> <td>Portrait is rotated 90 degrees to produce landscape. (For example, Hewlett-Packard PCL printers.)</td> </tr> <tr> <td>270</td> <td>Portrait is rotated 270 degrees to produce landscape. (For example, dot-matrix printers.)</td> </tr> </tbody> </table>	Value	Meaning	0	No landscape orientation.	90	Portrait is rotated 90 degrees to produce landscape. (For example, Hewlett-Packard PCL printers.)	270	Portrait is rotated 270 degrees to produce landscape. (For example, dot-matrix printers.)
Value	Meaning								
0	No landscape orientation.								
90	Portrait is rotated 90 degrees to produce landscape. (For example, Hewlett-Packard PCL printers.)								
270	Portrait is rotated 270 degrees to produce landscape. (For example, dot-matrix printers.)								
<u>DC_PAPER NAMES</u>	Retrieves a list of supported paper names--for example, Letter or Legal. If the lpszOutput parameter is NULL, the function returns the number of paper sizes available. Otherwise, lpszOutput points to an array for the paper names in the form char [cPaperNames, 64]. Each paper name is a null-terminated string.								
<u>DC_PAPERS</u>	Retrieves a list of supported paper sizes. The function copies the list to lpszOutput as a <u>WORD</u> array and returns the number of entries in the array. If lpszOutput is NULL, the function returns the number of supported paper sizes to allow the application the opportunity to allocate a buffer with the correct size. For more information on paper sizes, see the description of the dmPaperSize member of the <u>DEVMODE</u> structure.								
<u>DC_PAPERSIZE</u>	Copies the dimensions of all supported paper sizes, in tenths of a millimeter, to an array of <u>POINT</u> structures pointed to by the lpszOutput parameter. The width (x-dimension) and length (y-dimension) of a paper size are returned as if the paper were in the <u>DMORIENT_PORTRAIT</u> orientation.								
<u>DC_SIZE</u>	Returns the dmSize member of the printer driver's <u>DEVMODE</u> structure.								
<u>DC_TRUETYPE</u>	Retrieves the abilities of the driver to use								

TrueType fonts. The return value can be one or more of the following:

Value	Meaning
DCTT_BITMAP	Device is capable of printing TrueType fonts as graphics. (For example, dot-matrix and PCL printers.)
DCTT_DOWNLOAD	Device is capable of downloading TrueType fonts. (For example, PCL and PostScript printers.)
DCTT_SUBDEV	Device is capable of substituting device fonts for TrueType fonts. (For example, PostScript printers.)

For DC\_TRUETYPE, the `lpszOutput` parameter should be NULL.

#### DC\_VERSION

Returns the specification version to which the printer driver conforms.

`lpszOutput` Points to an array of bytes. The format of the array depends on the setting of the `fwCapability` parameter. If `lpszOutput` is zero, `DeviceCapabilities` returns the number of bytes required for the output data.

`lpdm` Points to a DEVMODE structure. If this parameter is NULL, `DeviceCapabilities` retrieves the current default initialization values for the specified printer driver. Otherwise, the function retrieves the values contained in the structure to which `lpdm` points.

#### Returns

The return value, if the function is successful, depends on the setting of the `fwCapability` parameter. The return value is -1 if the function fails.

#### Comments

This function is supplied by the printer driver. To use the `DeviceCapabilities` function, an application must retrieve the address of the function by calling the LoadLibrary and GetProcAddress functions, and it must include the `PRINT.H` file.

`DeviceCapabilities` is not supported by all printer drivers. If the GetProcAddress function returns NULL, `DeviceCapabilities` is not supported.

#### See Also

GetProcAddress, LoadLibrary



## Changes

The following index values have been added for Windows version 3.1:

DC\_COPIES  
DC\_ENUMRESOLUTIONS  
DC\_FILEDEPENDENCIES  
DC\_ORIENTATION  
DC\_PAPER NAMES  
DC\_TRUETYPE  
DCTT\_BITMAP  
DCTT\_DOWNLOAD  
DCTT\_SUBDEV

## Corrections

Changed the type of the last argument `lpdm` from `LPFNDEV MODE` to `LPDEV MODE`.

DC\_BINNAMES 12

Copies an array containing a list of the names of the paper bins. This array is in the form char PaperNames[cBinMax][cchBinName] where cchBinName is 24. If the lpszOutput parameter is NULL, the return value is the number of bin entries required. Otherwise, the return value is the number of bins copied.

DC\_BINNAMES 12

## DC\_BINS 6

Retrieves a list of available bins. The function copies the list to the `lpszOutput` parameter as a WORD array. If `lpszOutput` is `NULL`, the function returns the number of supported bins to allow the application the opportunity to allocate a buffer with the correct size. For more information about these bins, see the description of the `dmDefaultSource` member of the DEVMODE structure.

DC\_BINS 6

DC\_COPIES 18

Returns the number of copies the device can print.

DC\_COPIES 18

DC\_DRIVER 11

Returns the version number of the printer driver.



DC\_DRIVER 11

DC\_DUPLEX 7

Returns the level of duplex support. The function returns 1 if the printer is capable of duplex printing. Otherwise, the return value is zero.

DC\_DUPLEX 7

## DC\_ENUMRESOLUTIONS 13

Returns a list of available resolutions. If `lpszOutput` is `NULL`, the function returns the number of available resolution configurations. Resolutions are represented by pairs of LONG integers representing the horizontal and vertical resolutions (specified in dots per inch).



DC\_EXTRA 9

Returns the number of bytes required for the device-specific portion of the DEVMODE structure for the printer driver.

DC\_EXTRA 9

## DC\_FIELDS 1

Returns the dmFields member of the printer driver's DEVMODE structure. The dmFields member indicates which fields in the device-independent portion of the structure are supported by the printer driver.



DC\_FIELDS 1

#### DC\_FILEDEPENDENCIES 14

Returns a list of files that also need to be loaded when a driver is installed. If the `lpszOutput` parameter is `NULL`, the function returns the number of files. Otherwise, `lpszOutput` points to an array of filenames in the form `char[chFileName, 64]`. Each filename is a null-terminated string.



DC\_MAXEXTENT 5

Returns a POINT structure containing the maximum paper size that the dmPaperLength and dmPaperWidth members of the printer driver's DEVMODE structure can specify.

DC\_MAXEXTENT 5

DC\_MINEXTENT 4

Returns a POINT structure containing the minimum paper size that the dmPaperLength and dmPaperWidth members of the printer driver's DEVMODE structure can specify.

DC\_MINEXTENT 4

## DC\_ORIENTATION 17

Returns the relationship between portrait and landscape orientations for a device, in terms of the number of degrees that portrait orientation is rotated counterclockwise to produce landscape orientation. The return value can be one of the following:





## DC\_PAPER NAMES 16

Retrieves a list of supported paper names--for example, Letter or Legal. If the `lpszOutput` parameter is `NULL`, the function returns the number of paper sizes available. Otherwise, `lpszOutput` points to an array for the paper names in the form `char[cPaperNames, 64]`. Each paper name is a null-terminated string.

DC\_PAPER NAMES 16

## DC\_PAPERS 2

Retrieves a list of supported paper sizes. The function copies the list to `lpszOutput` as a WORD array and returns the number of entries in the array. If `lpszOutput` is `NULL`, the function returns the number of supported paper sizes to allow the application the opportunity to allocate a buffer with the correct size. For more information on paper sizes, see the description of the `dmPaperSize` member of the DEVMODE structure.

DC\_PAPERS 2

### DC\_PAPERSIZE 3

Copies the dimensions of all supported paper sizes, in tenths of a millimeter, to an array of POINT structures pointed to by the `lpszOutput` parameter. The width (x-dimension) and length (y-dimension) of a paper size are returned as if the paper were in the `DMORIENT_PORTRAIT` orientation.

DC\_PAPERSIZE 3

DC\_SIZE 8

Returns the dmSize member of the printer driver's DEVMODE structure.



DC\_SIZE 8

DC\_TRUETYPE 15

Retrieves the abilities of the driver to use TrueType fonts. The return value can be one or more of the following:

DC\_TRUETYPE 15

For  
DC\_TRUETYPE, the lpszOutput parameter should be NULL.

DC\_VERSION 10

Returns the specification version to which the printer driver conforms.

DC\_VERSION 10

## DeviceMode (2.x)

```
void DeviceMode(HWND hwnd, HMODULE hModule, LPCTSTR lpszDevice, LPCTSTR lpszOutput)
/* handle of window owning dialog box */
HANDLE hModule; /*
handle of printer-driver module
/
LPCTSTR lpszDevice;
* address of string for device name
/
LPCTSTR lpszOutput;
* address of string for output name
/
```

The DeviceMode function sets the current printing modes for a specified device by using a dialog box to prompt for those modes. An application calls DeviceMode to allow the user to change the printing modes of the corresponding device. DeviceMode copies the mode information to the environment block that is associated with the device and maintained by the graphics device interface (GDI).

The ExtDeviceMode function provides a superset of the functionality of the DeviceMode function; new applications should use ExtDeviceMode instead of DeviceMode whenever possible. (Applications can use the DM\_IN\_PROMPT constant with ExtDeviceMode to duplicate the functionality of DeviceMode.)

Parameter	Description
hwnd	Identifies the window that will own the dialog box.
hModule	Identifies the printer-driver module. The application should retrieve this handle by calling either the <u>GetModuleHandle</u> or <u>LoadLibrary</u> function.
lpszDevice	Points to a null-terminated string that specifies the name of the specific device to be supported (for example, Epson FX-80). The device name is the same as the name passed to the <u>CreateDC</u> function.
lpszOutput	Points to a null-terminated string that specifies the MS-DOS filename or device name for the physical output medium (file or output port). The output name is the same as the name passed to the <u>CreateDC</u> function.

### Returns

This function does not return a value.

### Comments

The DeviceMode function is part of the printer's device driver, not part of GDI. To call this function, an application must load the printer driver by calling the LoadLibrary function and retrieve the address of the function by using the GetProcAddress function. The application can then use the address to set up the printer.

DeviceMode is not supported by all printer drivers. If the GetProcAddress function returns NULL, DeviceMode is not supported.

### See Also

CreateDC, ExtDeviceMode, GetModuleHandle, LoadLibrary

## DPToLP (2.x)

```
BOOL DPToLP(hdc, lppt, cPoints)
HDC hdc;          /* handle of device context */
POINT FAR* lppt; /* address of
array with points
/
int cPoints;
* number of points in array
/
```

The DPToLP function converts device coordinates (points) into logical coordinates.

Parameter	Description
hdc	Identifies the device context.
lppt	Points to an array of <u>POINT</u> structures. Each coordinate in each structure is mapped into the logical coordinate system for the current device context.
cPoints	Specifies the number of points in the array.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The conversion depends on the current mapping mode and the settings of the origins and extents for the device's window and viewport.

### Example

The following example sets the mapping mode to MM\_LOENGLISH, and then calls the DPToLP function to convert the coordinates of a rectangle into logical coordinates:

```
RECT rc;
```

```
SetMapMode(hdc, MM_LOENGLISH);
SetRect(&rc, 100, 100, 200, 200);
DPToLP(hdc, (LPPOINT) &rc, 2);
```

### See Also

LPtoDP, POINT



## Ellipse (2.x)

```
BOOL Ellipse(hdc, nLeftRect, nTopRect, nRightRect, nBottomRect)
HDC hdc;          /* handle of device context */
int nLeftRect;
* x-coordinate upper-left corner bounding rectangle
/
int nTopRect;
* y-coordinate upper-left corner bounding rectangle
/
int nRightRect;
* x-coordinate lower-right corner bounding rectangle
/
int nBottomRect;
* y-coordinate lower-right corner bounding rectangle
/
```

The Ellipse function draws an ellipse. The center of the ellipse is the center of the specified bounding rectangle. The ellipse is drawn by using the current pen, and its interior is filled by using the current brush.

If either the width or the height of the bounding rectangle is zero, the function does not draw the ellipse.

Parameter	Description
hdc	Identifies the device context.
nLeftRect	Specifies the logical x-coordinate of the upper-left corner of the bounding rectangle.
nTopRect	Specifies the logical y-coordinate of the upper-left corner of the bounding rectangle.
nRightRect	Specifies the logical x-coordinate of the lower-right corner of the bounding rectangle.
nBottomRect	Specifies the logical y-coordinate of the lower-right corner of the bounding rectangle.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The figure drawn by this function extends up to but does not include the right and bottom coordinates. This means that the height of the figure is determined as follows:

$nBottomRect - nTopRect$

Similarly, the width of the figure is determined as follows:

$nRightRect - nLeftRect$

Both the width and the height of a rectangle must be greater than 2 units and less than 32,767 units.

### See Also

[Arc](#), [Chord](#), [RECT](#)

## EndDoc (3.1)

```
int EndDoc(hdc)
HDC hdc; /* handle of device context */
```

The EndDoc function ends a print job. This function replaces the ENDDOC printer escape for Windows version 3.1.

Parameter	Description
hdc	Identifies the device context for the print job.

### Returns

The return value is greater than or equal to zero if the function is successful. Otherwise, it is less than zero.

### Comments

An application should call the EndDoc function immediately after finishing a successful print job. To terminate a print job because of an error or if the user chooses to cancel the job, an application should call the AbortDoc function.

Do not use the EndDoc function inside metafiles.

### See Also

AbortDoc, Escape, StartDoc

## EndPage (3.1)

```
int EndPage(hdc)
HDC hdc; /* handle of device context */
```

The EndPage function signals the device that the application has finished writing to a page. This function is typically used to direct the driver to advance to a new page.

This function replaces the NEWFRAME printer escape for Windows 3.1. Unlike NEWFRAME, this function is always called after printing a page.

Parameter	Description
hdc	Identifies the device context for the print job.

### Returns

The return value is greater than or equal to zero if the function is successful. Otherwise, it is an error value.

### Errors

If the function fails, it returns one of the following error values:

Value	Meaning
<u>SP_ERROR</u>	General error.
<u>SP_APPABORT</u>	Job was terminated because the application's print-canceling function returned zero.
<u>SP_USERABORT</u>	User terminated the job by using Windows Print Manager (PRINTMAN.EXE).
<u>SP_OUTOFDISK</u>	Not enough disk space is currently available for spooling, and no more space will become available.
<u>SP_OUTOFMEMORY</u>	Not enough memory is available for spooling.

### Comments

The ResetDC function can be used to change the device mode, if necessary, after calling the EndPage function.

### See Also

Escape, ResetDC, StartPage

SP\_ERROR (-1)

General error.

SP\_ERROR (-1)

SP\_APPABORT (-2)

Job was terminated because the application's print-canceling function returned zero.

SP\_APPABORT (-2)

SP\_USERABORT (-3)

User terminated the job by using Windows Print Manager (PRINTMAN.EXE).



SP\_USERABORT (-3)

SP\_OUTOFDISK (-4)

Not enough disk space is currently available for spooling, and no more space will become available.

SP\_OUTOFDISK (-4)

SP\_OUTOFMEMORY (-5)

Not enough memory is available for spooling.

SP\_OUTOFMEMORY (-5)

## EnumFontFamilies (3.1)

```
int EnumFontFamilies(hdc, lpszFamily, fntenmprc, lParam)
HDC hdc; /* handle of device context */
LPCSTR lpszFamily; /* address of font-family name */
FONTENUMPROC fntenmprc; /* address of callback function */
LPARAM lParam; /* application-defined data */
```

The EnumFontFamilies function enumerates the fonts in a specified font family that are available on a given device. EnumFontFamilies continues until there are no more fonts or the callback function returns zero.

Parameter	Description
hdc	Identifies the device context.
lpszFamily	Points to a null-terminated string that specifies the family name of the desired fonts. If this parameter is NULL, the EnumFontFamilies function selects and enumerates one font from each available font family.
fntenmprc	Specifies the procedure-instance address of the application-defined callback function. The address must be created by the <a href="#">MakeProcInstance</a> function. For more information about the callback function, see the <a href="#">description of the EnumFontFamProc</a> callback function.
lParam	Specifies a 32-bit application-defined value that is passed to the callback function along with the font information.

### Returns

The return value specifies the last value returned by the callback function, if the function is successful. This value depends on which font families are available for the given device.

### Comments

The EnumFontFamilies function differs from the [EnumFonts](#) function in that it retrieves the style names associated with a TrueType font. Using [EnumFontFamilies](#), an application can retrieve information about unusual font styles (for example, Outline) that cannot be enumerated by using the [EnumFonts](#) function. Applications should use [EnumFontFamilies](#) instead of [EnumFonts](#).

For each font having the font name specified by the lpszFamily parameter, the EnumFontFamilies function retrieves information about that font and passes it to the function pointed to by the fntenmprc parameter. The application-supplied callback function can process the font information, as necessary.

### Example

The following example uses the [MakeProcInstance](#) function to create a pointer to the callback function for the EnumFontFamilies function. The [FreeProcInstance](#) function is called when enumeration is complete. Because the second parameter is NULL, EnumFontFamilies enumerates one font from each family that is available in the given device context. The aFontCount variable points to an array that is used inside the callback function.

```
FONTENUMPROC lpEnumFamCallBack;
int aFontCount[] = { 0, 0, 0 };

lpEnumFamCallBack = (FONTENUMPROC) MakeProcInstance(
    (FARPROC) EnumFamCallBack, hAppInstance);
EnumFontFamilies(hdc, NULL, lpEnumFamCallBack, (LPARAM) aFontCount);
FreeProcInstance((FARPROC) lpEnumFamCallBack);
```

### See Also

[EnumFonts](#), [EnumFontFamProc](#), [LOGFONT](#), [TEXTMETRIC](#)

## EnumFonts (2.x)

```
int EnumFonts(hdc, lpszFace, fntenmprc, lParam)
HDC hdc; /* handle of device context */
LPCSTR lpszFace; /* address of font name */
FONTENUMPROC fntenmprc; /* address of callback function */
LPARAM lParam; /* application-defined data */
```

The EnumFonts function enumerates the fonts available for a given device. This function is provided for backwards compatibility with earlier versions of Windows; current applications should use the [EnumFontFamilies](#) function.

EnumFonts continues until there are no more fonts or the callback function returns zero.

Parameter	Description
hdc	Identifies the device context.
lpszFace	Points to a null-terminated string that specifies the names of the requested fonts. If this parameter is NULL, the EnumFonts function randomly selects and enumerates one font from each available typeface.
fntenmprc	Specifies the procedure-instance address of the application-defined callback function. The address must be created by the <a href="#">MakeProcInstance</a> function. For more information about the callback function, see the <a href="#">description of the EnumFontsProc</a> callback function.
lParam	Specifies a 32-bit application-defined value that is passed to the callback function along with the font information.

### Returns

The return value specifies the last value returned by the callback function and is defined by the user.

### Comments

The EnumFonts function retrieves information about the specified font and passes it to the function pointed to by the fntenmprc parameter. The application-supplied callback function can process the font information, as necessary.

If the device is capable of text transformations (scaling, italicizing, and so on), only the base font will be enumerated. The user must know the device's text-transformation abilities to determine which additional fonts are available directly from the device. The graphics device interface ([GDI](#)) can simulate the bold, italic, underlined, and strikethrough attributes for any GDI-based font.

The EnumFonts function enumerates fonts from the [GDI](#) internal table only. This does not include fonts that are generated by a device, such as fonts that are transformations of fonts from the internal table. The [GetDeviceCaps](#) function can be used to determine which transformations a device can perform. This information is available by using the TEXTCAPS index.

[GDI](#) can scale GDI-based raster fonts by one to five units horizontally and one to eight units vertically, unless [PROOF\\_QUALITY](#) is being used.

### Example

The following example uses the [MakeProcInstance](#) function to create a pointer to the callback function for the EnumFonts function. The [FreeProcInstance](#) function is called when enumeration is complete. Because the second parameter is "Arial", EnumFonts enumerates the Arial fonts available in the given device context. The cArial variable is passed to the callback function.

```
FONTENUMPROC lpEnumFontsCallBack;
int cArial = 0;

lpEnumFontsCallBack = (FONTENUMPROC) MakeProcInstance(
    (FARPROC) EnumFontsCallBack, hAppInstance);
EnumFonts(hdc, "Arial", lpEnumFontsCallBack, (LPARAM) &cArial);
FreeProcInstance((FARPROC) lpEnumFontsCallBack);
```

See Also

EnumFontFamilies, EnumFontsProc



## EnumMetaFile (2.x)

```
BOOL EnumMetaFile(hdc, hmf, mfenmprc, lParam)
HDC hdc; /* handle of device context */
HLOCAL hmf; /* handle of
metafile
/
MFENUMPROC mfenmprc;
* address of callback function
/
LPARAM lParam;
* application-defined data
/
```

The EnumMetaFile function enumerates the metafile records in a given metafile. EnumMetaFile continues until there are no more graphics device interface (GDI) calls or the callback function returns zero.

Parameter	Description
hdc	Identifies the device context associated with the metafile.
hmf	Identifies the metafile.
	Note: The <u>HLOCAL</u> type for this parameter is incorrect in the WINDOWS.H file. The type of this parameter is actually <u>HMETAFILE</u> . Developers should cast this parameter to an <u>HLOCAL</u> type to avoid compiler warnings.
mfenmprc	Specifies the procedure-instance address of the application-supplied callback function. The address must be created by using the <u>MakeProcInstance</u> function. For more information about the callback function, see the description of the <u>EnumMetaFileProc</u> callback function.
lParam	Specifies a 32-bit application-defined value that is passed to the callback function along with the metafile information.

### Returns

The return value is nonzero if the callback function enumerates all the GDI calls in a metafile. Otherwise, it is zero.

### Comments

The EnumMetaFile function retrieves metafile records and passes them to a callback function. An application can modify the metafile record inside the callback function. The application can also use the PlayMetaFileRecord function inside the callback function; this is useful for very large metafiles, when using the PlayMetaFile function might be time-consuming.

### Example

The following example creates a dashed green pen and passes it to the callback function for the EnumMetaFile function. If the first element in the array of object handles is a handle, that handle is replaced by the handle of the green pen before the PlayMetaFileRecord function is called. (For this example, it is assumed that the table of object handles contains only one handle and that it is the handle of a pen.)

```
MFENUMPROC lpEnumMetaProc;
HPEN hpenGreen;

lpEnumMetaProc = (MFENUMPROC) MakeProcInstance(
    (FARPROC) EnumMetaFileProc, hAppInstance);
hpenGreen = CreatePen(PS_DASH, 1, RGB(0, 255, 0));
EnumMetaFile(hdc, hmf, lpEnumMetaProc, (LPARAM) &hpenGreen);
FreeProcInstance((FARPROC) lpEnumMetaProc);
DeleteObject(hpenGreen);
.
.
.

int FAR PASCAL EnumMetaFileProc(HDC hdc, HANDLETABLE FAR* lpHTable,
METARECORD FAR* lpMFR, int cObj, BYTE FAR* lpClientData)
{
```

```
    if (lpHTable->objectHandle[0] != 0)
lpHTable->objectHandle[0] = *(HPEN FAR *) lpClientData;
    PlayMetaFileRecord(hdc, lpHTable, lpMFR, cObj);

    return 1;
}
```

See Also

EnumMetaFileProc, MakeProcInstance, PlayMetaFile, PlayMetaFileRecord

## EnumObjects (2.x)

```
int EnumObjects(hdc, fnObjectType, goenmprc, lParam)
HDC hdc; /* handle of device context */
int fnObjectType; /* type
of object */
/
GOBJENUMPROC goenmprc;
* address of callback function
/
LPARAM lParam;
* application-defined data
/
```

The EnumObjects function enumerates the pens and brushes available in the given device context. For each object of a given type, the callback function is called with the information for that object. EnumObjects continues until there are no more objects or the callback function returns zero.

Parameter	Description						
hdc	Identifies the device context.						
fnObjectType	Specifies the object type. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>OBJ_BRUSH</u></td><td>Specifies a brush.</td></tr><tr><td><u>OBJ_PEN</u></td><td>Specifies a pen.</td></tr></tbody></table>	Value	Meaning	<u>OBJ_BRUSH</u>	Specifies a brush.	<u>OBJ_PEN</u>	Specifies a pen.
Value	Meaning						
<u>OBJ_BRUSH</u>	Specifies a brush.						
<u>OBJ_PEN</u>	Specifies a pen.						
goenmprc	Specifies the procedure-instance address of the application-supplied callback function. The address must be created by the <u>MakeProcInstance</u> function. For more information about the callback function, see the <u>description of the EnumObjectsProc</u> callback function.						
lParam	Specifies a 32-bit application-defined value that is passed to the callback function.						

### Returns

The return value specifies the last value returned by the callback function and is defined by the user.

### Example

The following example retrieves the number of horizontally hatched brushes and fills LOGBRUSH structures with information about each of them:

```
#define MAXBRUSHES 50

GOBJENUMPROC lpProcCallback;
HGLOBAL hglbl;
LPBYTE lpbCountBrush;

lpProcCallback = (GOBJENUMPROC) MakeProcInstance(
    (FARPROC) Callback, hinst);

hglbl = GlobalAlloc(GMEM_FIXED, sizeof(LOGBRUSH)
    * MAXBRUSHES);
lpbCountBrush = (LPBYTE) GlobalLock(hglbl);
*lpbCountBrush = 0;
EnumObjects(hdc, OBJ_BRUSH, lpProcCallback,
    (LPARAM) lpbCountBrush);

FreeProcInstance((FARPROC) lpProcCallback);

int FAR PASCAL Callback(LPLOGBRUSH lpLogBrush, LPBYTE pbData)
{
    /*
    * The pbData parameter contains the number of horizontally
    * hatched brushes; the lpDest parameter is set to follow the
    * byte reserved for pbData and the LOGBRUSH structures that
    * have been filled with brush information.
    */
}
```

```

    LPLOGBRUSH lpDest =
    (LPLOGBRUSH) (pbData + 1 + (*pbData * sizeof(LOGBRUSH)));

    if (lpLogBrush->lbStyle ==
    BS_HATCHED && /* if horiz hatch */
    lpLogBrush->lbHatch == HS_HORIZONTAL) {
        *lpDest++ = *lpLogBrush; /* fills structure with brush info */
        (*pbData) ++; /* increments brush count*/
        if (*pbData >= MAXBRUSHES)
            return 0;
    }

    return 1;
}

```

See Also

EnumObjectsProc, FreeProcInstance, GlobalAlloc, GlobalLock, MakeProcInstance, LOGBRUSH,  
LOGPEN

OBJ\_BRUSH 2  
Specifies a brush.

OBJ\_BRUSH 2

OBJ\_PEN 1  
Specifies a pen.

OBJ\_PEN 1



## EqualRgn (2.x)

```
BOOL EqualRgn(hrgnSrc1, hrgnSrc2)
HRGN hrgnSrc1;      /* handle of first region to test for equality */
HRGN hrgnSrc2;      /*
handle of second region to test for equality
/
```

The EqualRgn function determines whether two given regions are identical.

Parameter	Description
hrgnSrc1	Identifies the first region.
hrgnSrc2	Identifies the second region.

### Returns

The return value is nonzero if the two regions are equal. Otherwise, it is zero.

### Example

The following example uses the EqualRgn function to test the equality of a region against two other regions. In this case, hrgn2 is identical to hrgn1, but hrgn3 is not identical to hrgn1.

```
BOOL fEqual;
HRGN hrgn1, hrgn2, hrgn3;
LPSTR lpszEqual = "Regions are equal.";
LPSTR lpszNotEqual = "Regions are not equal.";

hrgn1 = CreateRectRgn(10, 10, 110, 110); /* 1 and 2 identical */
hrgn2 = CreateRectRgn(10, 10, 110, 110);
hrgn3 = CreateRectRgn(100, 100, 210, 210); /* same dimensions */

fEqual = EqualRgn(hrgn1, hrgn2);
if (fEqual)
    TextOut(hdc, 10, 10, lpszEqual, lstrlen(lpszEqual));
else
    TextOut(hdc, 10, 10, lpszNotEqual, lstrlen(lpszNotEqual));

fEqual = EqualRgn(hrgn1, hrgn3);
if (fEqual)
    TextOut(hdc, 10, 30, lpszEqual, lstrlen(lpszEqual));
else
    TextOut(hdc, 10, 30, lpszNotEqual, lstrlen(lpszNotEqual));

DeleteObject(hrgn1);
DeleteObject(hrgn2);
DeleteObject(hrgn3);
```

## Escape (2.x)

```
int Escape(hdc, nEscape, cbInput, lpszInData, lpvOutData)
HDC hdc; /* handle of device context */
int nEscape; /* specifies
escape function
/
int cbInput;
* size of structure for input
/
LPCSTR lpszInData;
* address of structure for input
/
void FAR* lpvOutData;
* address of structure for output
/
```

The Escape function allows applications to access capabilities of a particular device that are not directly available through the graphics device interface (GDI). Escape calls made by an application are translated and sent to the driver.

Parameter	Description
hdc	Identifies the device context.
nEscape	Specifies the escape function to be performed.
cbInput	Specifies the number of bytes of data pointed to by the lpszInData parameter.
lpszInData	Points to the input structure required for the specified escape.
lpvOutData	Points to the structure that receives output from this escape. This parameter should be NULL if no data is returned.

### Returns

The return value specifies the outcome of the function. It is greater than zero if the function is successful, except for the QUERYESCSUPPORT printer escape, which checks for implementation only. The return value is zero if the escape is not implemented. A return value less than zero indicates an error.

### Errors

If the function fails, the return value is one of the following:

Value	Meaning
SP_ERROR	General error.
SP_OUTOFDISK	Not enough disk space is currently available for spooling, and no more space will become available.
SP_OUTOFMEMORY	Not enough memory is available for spooling.
SP_USERABORT	User terminated the job through Print Manager.

## Changes

Windows version 3.1 introduces six new functions that supersede some printer escapes:

Function	Description
<u>AbortDoc</u>	Terminates a print job. Supersedes the ABORTDOC escape.
<u>EndDoc</u>	Ends a print job. Supersedes the ENDDOC escape.
<u>EndPage</u>	Ends a page. Supersedes the NEWFRAME escape. Unlike NEWFRAME, this function is always called after printing a page.
<u>SetAbortProc</u>	Sets the abort function for a print job. Supersedes the SETABORTPROC escape.
<u>StartDoc</u>	Starts a print job. Supersedes the STARTDOC escape.
<u>StartPage</u>	Prepares printer driver to receive data.

The ResetDC function is also new for Windows version 3.1. ResetDC updates a device context, allowing such new functionality as changing the paper orientation or paper bin within a single print job. This ability was not supported by an escape in previous versions of Windows.

For a complete list of the printer escapes under Windows version 3.0, and how support has changed for Windows 3.1, see the Printer escapes topic.

## Printer escapes

Escape	Description
ABORTDOC	Superseded by the <u>AbortDoc</u> function in Windows version 3.1.
BANDINFO	Obsolete in Windows version 3.1. Because all printer drivers for Windows version 3.1 and later set the text flag in every band, this escape is useful only for older printer drivers.
BEGIN_PATH	No changes for Windows version 3.1. This escape is specific to PostScript printers.
CLIP_TO_PATH	No changes for Windows version 3.1. This escape is specific to PostScript printers.
DEVICEDATA	Superseded in Windows version 3.1. Applications should use the <u>PASSTHROUGH</u> escape to achieve the same functionality.
DRAFTMODE	Superseded in Windows version 3.1. Applications can achieve the same functionality by setting the <u>dmPrintQuality</u> member of the <u>DEVMODE</u> structure to <u>DMRES_DRAFT</u> and passing <u>this structure</u> to the <u>CreateDC</u> function.
DRAWPATTERNRECT	No changes for Windows version 3.1.
ENABLEDUPLEX	Superseded in Windows version 3.1. Applications can achieve the same functionality by setting the <u>dmDuplex</u> member of the <u>DEVMODE</u> structure and passing this structure to the <u>CreateDC</u> function.
ENABLEPAIRKERNING	No changes for Windows version 3.1.
ENABLERELATIVEWIDTHS	No changes for Windows version 3.1.
ENDDOC	Superseded by the <u>EndDoc</u> function in Windows version 3.1.
END_PATH	No changes for Windows version 3.1. This escape is specific to PostScript printers.
ENUMPAPERBINS	Superseded in Windows version 3.1. Applications can use the <u>DeviceCapabilities</u> function to achieve the same functionality.
ENUMPAPERMETRICS	Superseded in Windows version 3.1. Applications can use the <u>DeviceCapabilities</u> function to achieve the same functionality.
EPSPRINTING	No changes for Windows version 3.1. This escape is specific to PostScript printers.
EXT_DEVICE_CAPS	Superseded in Windows version 3.1. Applications can use the <u>GetDeviceCaps</u> function to achieve the same functionality. This escape is specific to PostScript printers.
EXTTEXTOUT	Superseded in Windows version 3.1. Applications can use the <u>ExtTextOut</u> function to achieve the same functionality. This escape is not supported by the version 3.1 PCL driver.
FLUSHOUTPUT	Removed for Windows version 3.1.
GETCOLORTABLE	Removed for Windows version 3.1.
GETEXTENDEDTEXTMETRICS	No changes for Windows version 3.1. Support for this escape may change in future versions of Windows.
GETTEXTENTTABLE	Superseded in Windows version 3.1. Applications can use the <u>GetCharWidth</u> function to achieve the same functionality. This escape is not supported by the version 3.1 PCL or PSCRIPT drivers.
GETFACENAME	No changes for Windows version 3.1. This escape is specific to PostScript printers.
GETPAIRKERNTABLE	No changes for Windows version 3.1.

GETPHYSPAGESIZE	No changes for Windows version 3.1. Support for this escape may change in future versions of Windows.
GETPRINTINGOFFSET	No changes for Windows version 3.1. Support for this escape may change in future versions of Windows.
GETSCALINGFACTOR	No changes for Windows version 3.1. Support for this escape may change in future versions of Windows.
GETSETPAPERBINS	Superseded in Windows version 3.1. Applications can achieve the same functionality by calling the <u>DeviceCapabilities</u> function to find the number of paper bins, calling the <u>ExtDeviceMode</u> function to find the current bin, and then setting the <u>dmDefaultSource</u> member of the <u>DEVMODE</u> structure and passing this structure to the <u>CreateDC</u> function.
	GETSETPAPERBINS changes the paper bin only for the current device context. A new device context will use the system-default paper bin until the bin is explicitly changed for that device context.
GETSETPAPERMETRICS	Obsolete in Windows version 3.1. Applications can use the <u>DeviceCapabilities</u> and <u>ExtDeviceMode</u> functions to achieve the same functionality.
GETSETPAPERORIENT	Obsolete in Windows version 3.1. Applications can achieve the same functionality by setting the <u>dmOrientation</u> member of the <u>DEVMODE</u> structure and passing this structure to the <u>CreateDC</u> function. This escape is not supported by the Windows 3.1 PCL driver.
GETSETSSCREENPARAMS	No changes for Windows version 3.1.
GETTECHNOLOGY	No changes for Windows version 3.1. Support for this escape may change in future versions of Windows. This escape is not supported by the Windows 3.1 PCL driver.
GETTRACKKERNTABLE	No changes for Windows version 3.1.
GETVECTORBRUSHSIZE	No changes for Windows version 3.1. Support for this escape may change in future versions of Windows.
GETVECTORPENSIZE	No changes for Windows version 3.1. Support for this escape may change in future versions of Windows.
MFCOMMENT	No changes for Windows version 3.1.
NEWFRAME	No changes for Windows version 3.1. Applications should use the <u>StartPage</u> and <u>EndPage</u> functions instead of this escape. Support for this escape may change in future versions of Windows.
NEXTBAND	No changes for Windows version 3.1. Support for this escape may change in future versions of Windows.
PASSTHROUGH	No changes for Windows version 3.1.
QUERYESCAPESUPPORT	No changes for Windows version 3.1.
RESTORE_CTM	No changes for Windows version 3.1. This escape is specific to PostScript printers.
SAVE_CTM	No changes for Windows version 3.1. This escape is specific to PostScript printers.
SELECTPAPERSOURCE	Obsolete in Windows version 3.1. Applications can achieve the same functionality by using the <u>DeviceCapabilities</u> function.
SETABORTPROC	Superseded in Windows version 3.1 by the <u>SetAbortProc</u> function.
SETALLJUSTVALUES	No changes for Windows version 3.1. Support for this escape may change in future versions of Windows. This escape is not supported by the Windows 3.1 PCL driver.
SET_ARC_DIRECTION	No changes for Windows version 3.1. This escape is specific to PostScript printers.
SET_BACKGROUND_COLOR	No changes for Windows version 3.1. Applications should use the <u>SetBkColor</u> function instead of this escape.

SET_BOUNDS	Support for this escape may change in future versions of Windows. No changes for Windows version 3.1. This escape is specific to PostScript printers.
SETCOLORTABLE	No changes for Windows version 3.1. Support for this escape may change in future versions of Windows.
SETCOPYCOUNT	Superseded in Windows version 3.1. An application should call the <u>DeviceCapabilities</u> function, specifying <u>DC_COPIES</u> for the <u>nIndex</u> parameter, to find the maximum number of copies the device can make. Then the application can set the number of copies by passing to the <u>CreateDC</u> function a pointer to the <u>DEVMODE</u> structure.
SETKERNTRACK	No changes for Windows version 3.1.
SETLINECAP	No changes for Windows version 3.1. This escape is specific to PostScript printers.
SETLINEJOIN	No changes for Windows version 3.1. This escape is specific to PostScript printers.
SETMITERLIMIT	No changes for Windows version 3.1. This escape is specific to PostScript printers.
SET_POLY_MODE	No changes for Windows version 3.1. This escape is specific to PostScript printers.
SET_SCREEN_ANGLE	No changes for Windows version 3.1.
SET_SPREAD	No changes for Windows version 3.1.
STARTDOC	Superseded in Windows version 3.1. An application should call the <u>StartDoc</u> function instead of this escape.
TRANSFORM_CTM	No changes for Windows version 3.1. This escape is specific to PostScript printers.

## ExcludeClipRect (2.x)

```
int ExcludeClipRect(hdc, nLeftRect, nTopRect, nRightRect, nBottomRect)
HDC hdc;          /* handle of device context */
int nLeftRect;   /* x-
coordinate top-left corner of rectangle
/
int nTopRect;
* y-coordinate top-left corner of rectangle
/
int nRightRect;
* x-coordinate bottom-right corner of rectangle
/
int nBottomRect;
* y-coordinate bottom-right corner of rectangle
/
```

The ExcludeClipRect function creates a new clipping region that consists of the existing clipping region minus the specified rectangle.

Parameter	Description
hdc	Identifies the device context.
nLeftRect	Specifies the logical x-coordinate of the upper-left corner of the rectangle.
nTopRect	Specifies the logical y-coordinate of the upper-left corner of the rectangle.
nRightRect	Specifies the logical x-coordinate of the lower-right corner of the rectangle.
nBottomRect	Specifies the logical y-coordinate of the lower-right corner of the rectangle.

### Returns

The return value is SIMPLEREGION (region has no overlapping borders), COMPLEXREGION (region has overlapping borders), or NULLREGION (region is empty), if the function is successful. Otherwise, the return value is ERROR (no region is created).

### Comments

The width of the rectangle, specified by the absolute value of nRightRect - nLeftRect, must not exceed 32,767 units. This limit applies to the height of the rectangle as well.

### Example

The following example uses the ExcludeClipRect function to create a clipping region in the shape of a frame that is 20 units wide. The frame is painted red when the FillRect function is used to paint the client area.

```
RECT rc;
HRGN hrgn;
HBRUSH hbrRed;

GetClientRect(hwnd, &rc);
hrgn = CreateRectRgn(10, 10, 110, 110);
SelectClipRgn(hdc, hrgn);

ExcludeClipRect(hdc, 30, 30, 90, 90);

hbrRed = CreateSolidBrush(RGB(255, 0, 0));
FillRect(hdc, &rc, hbrRed);

DeleteObject(hbrRed);
DeleteObject(hrgn);
```

### See Also

CombineRgn

## ExtDeviceMode (3.0)

```
#include <print.h>

int ExtDeviceMode(HWND hwnd, LPDEVMODE lpdmOutput, LPSTR lpszDevice, LPSTR lpszPort, LPDEVMODE lpdmInput, LPSTR lpszProfile, WORD fwMode)
/* handle of window */
HANDLE hDriver;
/* handle of driver */
/
LPDEVMODE lpdmOutput;
/* address of structure for driver output */
/
LPSTR lpszDevice;
/* string for name of device */
/
LPSTR lpszPort;
/* string for name of port */
/
LPDEVMODE lpdmInput;
/* address of structure for driver input */
/
LPSTR lpszProfile;
/* string for profile filename */
/
WORD fwMode;
/* operations mask */
/
```

The ExtDeviceMode function retrieves or modifies device initialization information for a given printer driver or displays a driver-supplied dialog box for configuring the printer driver. Printer drivers that support device initialization by applications export ExtDeviceMode so that applications can call it.

Parameter	Description
hwnd	Identifies a window. If the application calls the ExtDeviceMode function to display a dialog box, the specified window is the parent window of the dialog box.
hDriver	Identifies the device-driver module. The <u>GetModuleHandle</u> function or <u>LoadLibrary</u> function returns a module handle.
lpdmOutput	Points to a <u>DEVMODE</u> structure. The driver writes the initialization information supplied in the <u>lpdmInput</u> parameter to this structure.
lpszDevice	Points to a null-terminated string that contains the name of the printer device--for example, PCL/HP LaserJet.
lpszPort	Points to a null-terminated string that contains the name of the port to which the device is connected--for example, LPT1.
lpdmInput	Points to a <u>DEVMODE</u> structure that supplies initialization information to the printer driver.
lpszProfile	Points to a null-terminated string that contains the name of the initialization file, where initialization information is recorded and read from. If this parameter is NULL, WIN.INI is the default initialization file.
fwMode	Specifies a mask of values that determines the operations the function performs. If this parameter is zero, the ExtDeviceMode function returns the number of bytes required by the printer driver's <u>DEVMODE</u> structure. Otherwise, the fwMode parameter can be one or more of the following values (to change the print settings, the application must specify at least one input value and one output value):

Value	Meaning
<u>DM_IN_BUFFER</u>	Input value. Before prompting, copying, or updating, this value merges the printer driver's current print settings with the settings in the <u>DEVMODE</u> structure identified by the <u>lpdmInput</u> parameter. The structure is updated only for those members indicated by the application in the <u>dmFields</u> member. This value is also defined as <u>DM_MODIFY</u> .
<u>DM_IN_PROMPT</u>	Input value. This value presents the printer driver's Print Setup dialog box and then changes the settings in the



<u>DM_OUT_BUFFER</u>	printer's <u>DEVMODE</u> structure to values specified by the user. This value is also defined as <u>DM_PROMPT</u> . Output value. This value writes the printer driver's current print settings (including private data) to the <u>DEVMODE</u> structure identified by the <u>lpdmOutput</u> parameter. The calling application must allocate a buffer sufficiently large to contain the information. If this bit is clear, <u>lpdmOutput</u> can be NULL. This value is also defined as <u>DM_COPY</u> .
<u>DM_OUT_DEFAULT</u>	Output value. This value updates graphics device interface (GDI)'s current printer environment and the WIN.INI file, using the contents of the printer driver's <u>DEVMODE</u> structure. Avoid using this value, because it permanently changes the print settings for all applications. This value is also defined as <u>DM_UPDATE</u> .

#### Returns

If the fwMode parameter is zero, the return value is the size of the buffer required to contain the printer driver initialization data. (Note that this buffer can be larger than a DEVMODE structure, if the printer driver appends private data to the structure.) If the function displays the initialization dialog box, the return value is either IDOK or IDCANCEL, depending on which button the user selects. If the function does not display the dialog box and is successful, the return value is IDOK. The return value is less than zero if the function fails.

#### Comments

The ExtDeviceMode function is part of the printer's device driver and not part of GDI. To use this function, an application must retrieve the address of the function by calling the LoadLibrary and GetProcAddress functions, and it must include the header file PRINT.H. The application can then use the address to set up the printer.

ExtDeviceMode is not supported by all printer drivers. If the GetProcAddress function returns NULL, ExtDeviceMode is not supported.

To make changes to print settings that are local to the application, an application should call the ExtDeviceMode function, specifying the DM\_OUT\_BUFFER value; modify the returned DEVMODE structure; and then pass the modified DEVMODE structure back to ExtDeviceMode, specifying DM\_IN\_BUFFER and DM\_OUT\_BUFFER (combined by using the OR operator). The DEVMODE structure returned by this second call to ExtDeviceMode can be used as an argument in a call to the CreateDC function.

Any call to ExtDeviceMode must set either DM\_OUT\_BUFFER or DM\_OUT\_DEFAULT.

An application can set the fwMode parameter to DM\_OUT\_BUFFER to obtain a DEVMODE structure filled with the printer driver's initialization data. The application can then pass this structure to the CreateDC function to set a private environment for the printer device context.

#### See Also

CreateDC, DeviceMode, GetModuleHandle, GetProcAddress, LoadLibrary, DEVMODE

DM\_IN\_BUFFER DM\_MODIFY

Input value. Before prompting, copying, or updating, this value merges the printer driver's current print settings with the settings in the DEVMODE structure identified by the lpdmInput parameter. The structure is updated only for those members indicated by the application in the dmFields member. This value is also defined as DM\_MODIFY.

DM\_IN\_BUFFER DM\_MODIFY

DM\_IN\_PROMPT DM\_PROMPT

Input value. This value presents the printer driver's Print Setup dialog box and then changes the settings in the printer's DEVMODE structure to values specified by the user. This value is also defined as DM\_PROMPT.

DM\_IN\_PROMPT DM\_PROMPT

DM\_OUT\_BUFFER DM\_COPY

Output value. This value writes the printer driver's current print settings (including private data) to the DEVMODE structure identified by the `lpdmOutput` parameter. The calling application must allocate a buffer sufficiently large to contain the information. If this bit is clear, `lpdmOutput` can be NULL. This value is also defined as `DM_COPY`.

DM\_OUT\_BUFFER DM\_COPY

DM\_OUT\_DEFAULT DM\_UPDATE

Output value. This value updates graphics device interface (GDI)'s current printer environment and the WIN.INI file, using the contents of the printer driver's DEVMODE structure. Avoid using this value, because it permanently changes the print settings for all applications. This value is also defined as DM\_UPDATE.



DM\_OUT\_DEFAULT DM\_UPDATE

## ExtFloodFill (3.0)

```
BOOL ExtFloodFill(hdc, nXStart, nYStart, clrref, fuFillType)
HDC hdc; /* handle of device context */
int nXStart; /* x-coordinate
where filling begins */
int nYStart; /* y-coordinate where filling begins */
COLORREF clrref; /* color of fill */
UINT fuFillType; /* fill type */
```

The ExtFloodFill function fills an area of the screen surface by using the current brush. The type of flood fill specified determines which part of the screen is filled.

Parameter	Description						
hdc	Identifies the device context.						
nXStart	Specifies the logical x-coordinate at which to begin filling.						
nYStart	Specifies the logical y-coordinate at which to begin filling.						
clrref	Specifies the color of the boundary or area to be filled. The interpretation of this parameter depends on the value of the fuFillType parameter.						
fuFillType	Specifies the type of flood fill to be performed. It must be one of the following values:						
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>FLOODFILLBORDER</u></td><td>Fill area is bounded by the color specified by the clrref parameter. This style is identical to the filling performed by the FloodFill function.</td></tr><tr><td><u>FLOODFILLSURFACE</u></td><td>Fill area is defined by the color specified by the clrref parameter. Filling continues outward in all directions as long as the color is encountered. This style is useful for filling areas that have multicolored boundaries.</td></tr></tbody></table>	Value	Meaning	<u>FLOODFILLBORDER</u>	Fill area is bounded by the color specified by the clrref parameter. This style is identical to the filling performed by the FloodFill function.	<u>FLOODFILLSURFACE</u>	Fill area is defined by the color specified by the clrref parameter. Filling continues outward in all directions as long as the color is encountered. This style is useful for filling areas that have multicolored boundaries.
Value	Meaning						
<u>FLOODFILLBORDER</u>	Fill area is bounded by the color specified by the clrref parameter. This style is identical to the filling performed by the FloodFill function.						
<u>FLOODFILLSURFACE</u>	Fill area is defined by the color specified by the clrref parameter. Filling continues outward in all directions as long as the color is encountered. This style is useful for filling areas that have multicolored boundaries.						

### Returns

The return value is nonzero if the function is successful. It is zero if the filling cannot be completed, if the given point has the boundary color specified by the clrref parameter (if FLOODFILLBORDER was requested), if the given point does not have the color specified by clrref (if FLOODFILLSURFACE was requested), or if the point is outside the clipping region.

### Comments

Only memory device contexts and devices that support raster-display technology support the ExtFloodFill function.

If the fuFillType parameter is the FLOODFILLBORDER value, the area is assumed to be completely bounded by the color specified by the clrref parameter. The ExtFloodFill function begins at the coordinates specified by the nXStart and nYStart parameters and fills in all directions to the color boundary.

If fuFillType is FLOODFILLSURFACE, ExtFloodFill begins at the coordinates specified by nXStart and nYStart and continues in all directions, filling all adjacent areas containing the color specified by clrref.

### See Also

FloodFill, GetDeviceCaps

FLOODFILLBORDER 0

Fill area is bounded by the color specified by the clrref parameter. This style is identical to the filling performed by the FloodFill function.

FLOODFILLBORDER 0

## FLOODFILLSURFACE 1

Fill area is defined by the color specified by the clrref parameter. Filling continues outward in all directions as long as the color is encountered. This style is useful for filling areas that have multicolored boundaries.

FLOODFILLSURFACE 1

## ExtTextOut (2.x)

```
BOOL ExtTextOut(hdc, nXStart, nYStart, fuOptions, lprc, lpszString, cbString, lpDx)
HDC hdc; /* handle of device context */ /* x-coordinate
int nXStart;
of starting position
/
int nYStart;
* y-coordinate of starting position
/
UINT fuOptions;
* rectangle type
/
const RECT FAR* lprc;
* address of structure with rectangle
/
LPCSTR lpszString;
* address of string
/
UINT cbString;
* number of bytes in string
/
int FAR* lpDx;
* spacing between character cells
/
```

The ExtTextOut function writes a character string within a rectangular region, using the currently selected font. The rectangular region can be opaque (filled by using the current background color as set by the SetBkColor function), and it can be a clipping region.

Parameter	Description						
hdc	Identifies the device context.						
nXStart	Specifies the logical x-coordinate at which the string begins.						
nYStart	Specifies the logical y-coordinate at which the string begins.						
fuOptions	Specifies the rectangle type. This parameter can be one, both, or neither of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>ETO_CLIPPED</u></td><td>Text is clipped to the rectangle.</td></tr><tr><td><u>ETO_OPAQUE</u></td><td>Current background color fills the rectangle. (An application can set and query the current background color by using the <u>SetBkColor</u> and <u>GetBkColor</u> functions.)</td></tr></tbody></table>	Value	Meaning	<u>ETO_CLIPPED</u>	Text is clipped to the rectangle.	<u>ETO_OPAQUE</u>	Current background color fills the rectangle. (An application can set and query the current background color by using the <u>SetBkColor</u> and <u>GetBkColor</u> functions.)
Value	Meaning						
<u>ETO_CLIPPED</u>	Text is clipped to the rectangle.						
<u>ETO_OPAQUE</u>	Current background color fills the rectangle. (An application can set and query the current background color by using the <u>SetBkColor</u> and <u>GetBkColor</u> functions.)						
lprc	Points to a <u>RECT</u> structure that determines the dimensions of the rectangle.						
lpszString	Points to the specified character string.						
cbString	Specifies the number of bytes in the string.						
lpDx	Points to an array of values that indicate the distance, in logical units, between origins of adjacent character cells. The nth element in the array specifies the number of logical units that separate the origin of the nth item in the string from the origin of item n + 1. If this parameter is NULL, ExtTextOut uses the default spacing between characters. Otherwise, the array contains the number of elements specified in the cbString parameter.						

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

If the fuOptions parameter is zero and the lprc parameter is NULL, the ExtTextOut function writes text to the device context without using a rectangular region.

By default, the current position is not used or updated by ExtTextOut. If an application needs to update the current position when it calls ExtTextOut, the application can call the SetTextAlign function with the wFlags parameter set to TA\_UPDATECP. When this flag is set, Windows ignores the nXStart and nYStart parameters on subsequent calls to ExtTextOut, using the current position instead. When an application

uses TA\_UPDATECP to update the current position, ExtTextOut sets the current position either to the end of the previous line of text or to the position specified by the last element of the array pointed to by the lpDX parameter, whichever is greater.

Example

The following example uses the ExtTextOut function to clip text to a rectangular region defined by a RECT structure:

```
RECT rc;
```

```
SetRect(&rc, 90, 190, 250, 220);
```

```
ExtTextOut(hdc, 100, 200, /* x and y coordinates */  
           ETO_CLIPPED, /* clips text to rectangle */  
           &rc, /* address of RECT structure */  
           "Test of ExtTextOut function.", /* string to write */  
           28, /* characters in string */  
           (LPINT) NULL); /* default character spacing */
```

See Also

GetBkColor, SetBkColor, SetTextAlign, SetTextColor, TabbedTextOut, TextOut, RECT



ETO\_CLIPPED 0x0004

Text is clipped to the rectangle.

ETO\_CLIPPED 0x0004

ETO\_OPAQUE 0x0002

Current background color fills the rectangle. (An application can set and query the current background color by using the SetBkColor and GetBkColor functions.)

ETO\_OPAQUE 0x0002

## FillRgn (2.x)

```
BOOL FillRgn(hdc, hrgn, hbr)
HDC hdc;          /* handle of device context */
HRGN hrgn;        /* handle of region */
HBRUSH hbr;      /* handle of brush */
/
```

The FillRgn function fills the given region by using the specified brush.

Parameter	Description
hdc	Identifies the device context.
hrgn	Identifies the region to be filled. The coordinates for the given region are specified in device units.
hbr	Identifies the brush to be used to fill the region.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Example

The following example uses a blue brush to fill a rectangular region. Note that it is not necessary to select the brush into the device context before using it to fill the region.

```
HRGN hrgn;
HBRUSH hBrush;

hrgn = CreateRectRgn(10, 10, 110, 110);
SelectObject(hdc, hrgn);

hBrush = CreateSolidBrush(RGB(0, 0, 255));

FillRgn(hdc, hrgn, hBrush);

DeleteObject(hrgn);
```

### See Also

CreateBrushIndirect, CreateDIBPatternBrush, CreateHatchBrush, CreatePatternBrush, CreateSolidBrush, PaintRgn

## FloodFill (2.x)

```
BOOL FloodFill(hdc, nXStart, nYStart, clrref)
HDC hdc;          /* handle of device context */
int nXStart;     /* x-coordinate of starting
position */
int nYStart;     /* y-coordinate of starting position */
COLORREF clrref; /* color of fill boundary */
```

The FloodFill function fills an area of the screen surface by using the current brush. The area is assumed to be bounded as specified by the clrref parameter. The FloodFill function begins at the point specified by the nXStart and nYStart parameters and continues in all directions to the color boundary.

Parameter	Description
hdc	Identifies the device context.
nXStart	Specifies the logical x-coordinate at which to begin filling.
nYStart	Specifies the logical y-coordinate at which to begin filling.
clrref	Specifies the color of the boundary.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero, indicating that the filling cannot be completed, that the given point has the boundary color specified by clrref, or that the point is outside the clipping region.

### Comments

Only memory device contexts and devices that support raster-display technology support the FloodFill function.

### See Also

[ExtFloodFill](#), [GetDeviceCaps](#)

## FrameRgn (2.x)

```
BOOL FrameRgn(hdc, hrgn, hbr, nWidth, nHeight)
HDC hdc;          /* handle of device context */
HRGN hrgn;        /* handle of region */
HBRUSH hbr;
* handle of brush
/
int nWidth;
* width of region frame
/
int nHeight;
* height of region frame
/
```

The FrameRgn function draws a border around the given region, using the specified brush.

Parameter	Description
hdc	Identifies the device context.
hrgn	Identifies the region to be enclosed in a border.
hbr	Identifies the brush to be used to draw the border.
nWidth	Specifies the width, in device units, of vertical brush strokes.
nHeight	Specifies the height, in device units, of horizontal brush strokes.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Example

The following example uses a blue brush to frame a rectangular region. Note that it is not necessary to select the brush or the region into the device context.

```
HRGN hrgn;
HBRUSH hBrush;
int Width = 5, Height = 2;

hrgn = CreateRectRgn(10, 10, 110, 110);
hBrush = CreateSolidBrush(RGB(0, 0, 255));

FrameRgn(hdc, hrgn, hBrush, Width, Height);

DeleteObject(hrgn);
DeleteObject(hBrush);

See Also
FillRgn, PaintRgn
```

## GetAspectRatioFilter (2.x)

```
DWORD GetAspectRatioFilter(hdc)  
HDC hdc; /* handle of device context */
```

The `GetAspectRatioFilter` function retrieves the setting for the current aspect-ratio filter. The aspect ratio is the ratio formed by a device's pixel width and height. Information about a device's aspect ratio is used in the creation, selection, and display of fonts. Windows provides a special filter, the aspect-ratio filter, to select fonts designed for a particular aspect ratio from all of the available fonts. The filter uses the aspect ratio specified by the [SetMapperFlags](#) function.

Parameter	Description
<code>hdc</code>	Identifies the device context that contains the specified aspect ratio.

### Returns

The low-order word of the return value contains the x-coordinate of the aspect ratio if the function is successful; the high-order word contains the y-coordinate.

### See Also

[SetMapperFlags](#)



### GetAspectRatioFilterEx (3.1)

```
BOOL GetAspectRatioFilterEx(hdc, lpAspectRatio)
HDC hdc;
SIZE FAR* lpAspectRatio;
```

The `GetAspectRatioFilterEx` function retrieves the setting for the current aspect-ratio filter. The aspect ratio is the ratio formed by a device's pixel width and height. Information about a device's aspect ratio is used in the creation, selection, and displaying of fonts. Windows provides a special filter, the aspect-ratio filter, to select fonts designed for a particular aspect ratio from all of the available fonts. The filter uses the aspect ratio specified by the [SetMapperFlags](#) function.

Parameter	Description
<code>hdc</code>	Identifies the device context that contains the specified aspect ratio.
<code>lpAspectRatio</code>	Pointer to a <a href="#">SIZE</a> structure where the current aspect ratio filter will be returned.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### See Also

[SetMapperFlags](#)

## GetBitmapBits (2.x)

```
LONG GetBitmapBits(hbm, cbBuffer, lpvBits)
HBITMAP hbm;          /* handle of bitmap */
LONG cbBuffer;        /* number of
bytes to copy to buffer
/
void FAR* lpvBits;
* address of buffer for bitmap bits
/
```

The GetBitmapBits function copies the bits of the specified bitmap into a buffer.

Parameter	Description
hbm	Identifies the bitmap.
cbBuffer	Specifies the number of bytes to be copied.
lpvBits	Points to the buffer that is to receive the bitmap. The bitmap is an array of bytes. This array conforms to a structure in which horizontal scan lines are multiples of 16 bits.

### Returns

The return value specifies the number of bytes in the bitmap if the function is successful. It is zero if there is an error.

### Comments

An application can use the GetObject function to determine the number of bytes to copy into the buffer pointed to by the lpvBits parameter.

### See Also

GetObject, SetBitmapBits

## GetBitmapDimension (2.x)

```
DWORD GetBitmapDimension(hbm)
HBITMAP hbm;          /* handle of bitmap */
```

The GetBitmapDimension function returns the width and height of the specified bitmap. The height and width is assumed to have been set by the SetBitmapDimension function.

Parameter	Description
hbm	Identifies the bitmap.

### Returns

The low-order word of the return value contains the bitmap width, in tenths of a millimeter, if the function is successful; the high-order word contains the height. If the bitmap width and height have not been set by using the SetBitmapDimension function, the return value is zero.

### See Also

SetBitmapDimension

## GetBitmapDimensionEx (2.x)

```
BOOL GetBitmapDimensionEx(hBitmap, lpDimension)
HBITMAP hBitmap;          /* handle of bitmap */
SIZE FAR* lpDimension;    /* address of dimension structure */
/
```

The GetBitmapDimensionEx function returns the dimensions of the bitmap previously set by the SetBitmapDimensionEx function. If no dimensions have been set, a default of 0,0 will be returned.

Parameter	Description
hBitmap	Identifies the bitmap.
lpDimension	Points to a <u>SIZE</u> structure to which the dimensions are returned.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### See Also

SetBitmapDimensionEx, SIZE

## GetBkColor (2.x)

COLORREF GetBkColor(hdc)  
HDC hdc; /\* handle of device context \*/

The GetBkColor function returns the current background color.

Parameter	Description
hdc	Identifies the device context.

### Returns

The return value is an RGB (red, green, blue) color value if the function is successful.

### Comments

If the background mode is OPAQUE, the system uses the background color to fill the gaps in styled lines, the gaps between hatched lines in brushes, and the background in character cells. The system also uses the background color when converting bitmaps between color and monochrome device contexts.

### Example

The following example uses the GetBkColor function to determine whether the current background color is white. If it is, the SetBkColor function sets it to red.

```
DWORD dwBackColor;
```

```
dwBackColor = GetBkColor(hdc);  
if (dwBackColor == RGB(255, 255, 255)) { /* if color is white */  
    SetBkColor(hdc, RGB(255, 0, 0)); /* sets color to red */  
    TextOut(hdc, 100, 200, "SetBkColor test.", 16);  
}
```

### See Also

GetBkMode, SetBkColor, SetBkMode, RGB

## GetBkMode (2.x)

```
int GetBkMode(hdc)
HDC hdc;      /* handle of device context */
```

The `GetBkMode` function returns the background mode. The background mode defines whether the system removes existing background colors on the drawing surface before drawing text, hatched brushes, or any pen style that is not a solid line.

Parameter	Description
<code>hdc</code>	Identifies the device context.

### Returns

The return value specifies the current background mode if the function is successful. It can be [OPAQUE](#), [TRANSPARENT](#), or [TRANSPARENT1](#).

### Example

The following example determines the current background mode by calling the `GetBkMode` function. If the mode is [OPAQUE](#), the [SetBkMode](#) function sets it to [TRANSPARENT](#).

```
int nBackMode;

nBackMode = GetBkMode(hdc);
if (nBackMode == OPAQUE) {
    TextOut(hdc, 90, 100, "This background mode is OPAQUE.", 31);
    SetBkMode(hdc, TRANSPARENT);
}
```

### See Also

[GetBkColor](#), [SetBkColor](#), [SetBkMode](#)

### GetBoundsRect (3.1)

UINT GetBoundsRect(hdc, lprcBounds, flags)

HDC hdc; /\* handle of device context \*/

RECT FAR\* lprcBounds;

\* address of structure for bounding rectangle

/

UINT flags;

\* specifies whether to clear rectangle

/

The GetBoundsRect function returns the current accumulated bounding rectangle for the specified device context.

Parameter	Description
hdc	Identifies the device context to return the bounding rectangle for.
lprcBounds	Points to a buffer that will receive the current bounding rectangle. The rectangle is returned in logical coordinates.
flags	Specifies whether the bounding rectangle to be cleared after it is returned. This parameter can be DCB_RESET, to clear the rectangle. Otherwise, it should be zero.

#### Returns

The return value is DCB\_SET if the bounding rectangle is not empty. Otherwise it is DCB\_RESET.

#### See Also

SetBoundsRect

## GetBrushOrg (2.x)

```
DWORD GetBrushOrg(hdc)
HDC hdc;      /* handle of device context */
```

The GetBrushOrg function retrieves the origin, in device coordinates, of the brush currently selected for the given device context.

Parameter	Description
hdc	Identifies the device context.

### Returns

The low-order word of the return value contains the current x-coordinate of the brush, in device coordinates, if the function is successful; the high-order word contains the y-coordinate.

### Comments

The initial brush origin is at the coordinates (0,0) in the client area. The return value specifies these coordinates in device units relative to the origin of the desktop window.

### Example

The following example uses the LOWORD and HIWORD macros to extract the x- and y-coordinate of the current brush from the return value of the GetBrushOrg function:

```
DWORD dwBrOrg;
WORD  wXBrOrg, wYBrOrg;
```

```
dwBrOrg = GetBrushOrg(hdc);
wXBrOrg = LOWORD(dwBrOrg);
wYBrOrg = HIWORD(dwBrOrg);
```

### See Also

GetBrushOrgEx, SelectObject, SetBrushOrg, HIWORD, LOWORD



### GetBrushOrgEx (3.1)

```
BOOL GetBrushOrgEx(hDC, lpPoint)
HDC hDC;          /* handle of device context */
POINT FAR* lpPoint;
* address of structure for brush origin
/
```

The GetBrushOrgEx function retrieves the current brush origin for the given device context.

Parameter	Description
hDC	Identifies the device context.
lpPoint	Points to a <u>POINT</u> structure to which the device coordinates of the brush origin are to be returned.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The initial brush origin is at the coordinate (0,0).

#### See Also

GetBrushOrg, SetBrushOrg

### GetCharABCWidths (3.1)

```
BOOL GetCharABCWidths(hdc, uFirstChar, uLastChar, lpabc)
HDC hdc;          /* handle of device context */
UINT uFirstChar; /* first character
in range to query
/
UINT uLastChar;
* last character in range to query
/
LPABC lpabc;
* address of ABC width structures
/
```

The `GetCharABCWidths` function retrieves the widths of consecutive characters in a specified range from the current TrueType font. The widths are returned in logical units. This function succeeds only with TrueType fonts.

Parameter	Description
<code>hdc</code>	Identifies the device context.
<code>uFirstChar</code>	Specifies the first character in the range of characters from the current font for which character widths are returned.
<code>uLastChar</code>	Specifies the last character in the range of characters from the current font for which character widths are returned.
<code>lpabc</code>	Points to an array of <code>ABC</code> structures that receive the character widths when the function returns. This array must contain at least as many <code>ABC</code> structures as there are characters in the range specified by the <code>uFirstChar</code> and <code>uLastChar</code> parameters.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The TrueType rasterizer provides `ABC` character spacing after a specific point size has been selected. "A" spacing is the distance that is added to the current position before placing the glyph. "B" spacing is the width of the black part of the glyph. "C" spacing is added to the current position to account for the white space to the right of the glyph. The total advanced width is given by  $A + B + C$ .

When the `GetCharABCWidths` function retrieves negative "A" or "C" widths for a character, that character includes underhangs or overhangs.

To convert the `ABC` widths to font design units, an application should create a font whose height (as specified in the `lFHeight` member of the `LOGFONT` structure) is equal to the value stored in the `ntmSizeEM` member of the `NEWTEXTMETRIC` structure. (The value of the `ntmSizeEM` member can be retrieved by calling the `EnumFontFamilies` function.)

The `ABC` widths of the default character are used for characters that are outside the range of the currently selected font.

To retrieve the widths of characters in non-TrueType fonts, applications should use the `GetCharWidth` function.

#### See Also

[EnumFontFamilies](#), [GetCharWidth](#), [ABC](#), [OUTLINETEXTMETRIC](#)

## GetCharWidth (2.x)

```
BOOL GetCharWidth(hdc, uFirstChar, uLastChar, lpnWidths)
HDC hdc; /* handle of device context */
UINT uFirstChar; /* first character
in range to query
/
UINT uLastChar;
* last character in range to query
/
int FAR* lpnWidths;
* address of buffer for widths
/
```

The GetCharWidth function retrieves the widths of individual characters in a range of consecutive characters in the current font.

Parameter	Description
hdc	Identifies the device context.
uFirstChar	Specifies the first character in a group of consecutive characters in the current font.
uLastChar	Specifies the last character in a group of consecutive characters in the current font.
lpnWidths	Points to a buffer that receives the width values for a group of consecutive characters in the current font.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

If a character in the group of consecutive characters does not exist in a particular font, it will be assigned the width value of the default character.

### Example

The following example uses the GetCharWidth function to retrieve the widths of the characters from "I" through "S" and displays the total number of widths retrieved in a message box:

```
HDC hdc;
WORD wTotalValues;
WORD wFirstChar, wLastChar;
int InfoBuffer[256];
char szMessage[30];

wFirstChar = (WORD) 'I';
wLastChar = (WORD) 'S';

hdc = GetDC(hwnd);
if (GetCharWidth(hdc, wFirstChar, wLastChar, (int FAR*) InfoBuffer)) {
    wTotalValues = wLastChar - wFirstChar + 1;
    wsprintf(szMessage, "Total values received: %d", wTotalValues);
    MessageBox(hwnd, szMessage, "GetCharWidth", MB_OK);
}
else
    MessageBox(hwnd, "GetCharWidth was unsuccessful", "ERROR!",
    MB_OK);

ReleaseDC(hwnd, hdc);
```

### See Also

GetCharABCWidths

## GetClipBox (2.x)

```
int GetClipBox(hdc, lprc)
HDC hdc;          /* handle of device context */
RECT FAR* lprc;  /* address of
structure with rectangle
/
```

The GetClipBox function retrieves the dimensions of the smallest rectangle that completely contains the current clipping region.

Parameter	Description
hdc	Identifies the device context.
lprc	Points to the <u>RECT</u> structure that receives the logical coordinates of the rectangle.

### Returns

The return value is SIMPLEREGION (region has no overlapping borders), COMPLEXREGION (region has overlapping borders), or NULLREGION (region is empty), if the function is successful. Otherwise, the return value is ERROR.

### See Also

GetBoundsRect, GetRgnBox, GetTextExtent, SelectClipRgn, RECT

## GetCurrentPosition (2.x)

```
DWORD GetCurrentPosition(hdc)
HDC hdc; /* handle of device context */
```

The `GetCurrentPosition` function retrieves the logical coordinates of the current position. The current position is set by using the [MoveTo](#) function.

Parameter	Description
<code>hdc</code>	Identifies the device context.

### Returns

The low-order word of the return value contains the logical x-coordinate of the current position if the function is successful; the high-order word contains the logical y-coordinate.

### See Also

[GetCurrentPositionEx](#), [LineTo](#), [MoveTo](#)

### GetCurrentPositionEx (3.1)

BOOL GetCurrentPositionEx(hdc, lpPoint)

HDC hdc;

POINT FAR\* lpPoint;

The GetCurrentPositionEx function retrieves the current position in logical coordinates.

Parameter	Description
hdc	Identifies the device context to get the current position from.
lpPoint	Points to a <u>POINT</u> structure that gets filled with the current position.

#### Returns

The return value is nonzero if the function is successful, zero if there is an error.

#### See Also

GetCurrentPosition

## GetDCOrg (2.x)

```
DWORD GetDCOrg(hdc)
HDC hdc; /* handle of device context */
```

The GetDCOrg function retrieves the coordinates of the final translation origin for the device context. This origin specifies the offset used by Windows to translate device coordinates into client coordinates for points in an application's window. The final translation origin is relative to the physical origin of the screen.

Parameter	Description
hdc	Identifies the device context whose origin is to be retrieved.

### Returns

The low-order word of the return value contains the x-coordinate of the final translation origin, in device coordinates, if the function is successful; the high-order word contains the y-coordinate.

### Example

The following example uses the CreateIC function to create an information context for the screen and then retrieves the context's origin by using the GetDCOrg function:

```
HDC hdcIC;
DWORD dwOrigin;

hdcIC = CreateIC("DISPLAY", NULL, NULL, NULL);
dwOrigin = GetDCOrg(hdcIC);
```

```
DeleteDC(hdcIC);
```

See Also

CreateIC

## GetDeviceCaps (2.x)

```
int GetDeviceCaps(hdc, iCapability)
HDC hdc; /* handle of device context */
int iCapability; /* index of capability to query */
```

The GetDeviceCaps function retrieves device-specific information about a given display device.

Parameter	Description																																																																										
hdc	Identifies the device context.																																																																										
iCapability	Specifies the type of information to be returned. It can be one of the following indices:																																																																										
	<table border="1"> <thead> <tr> <th>Index</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>DRIVERVERSION</td> <td>Version number of the device driver.</td> </tr> <tr> <td>TECHNOLOGY</td> <td>Device technology. It can be one of the following values:</td> </tr> <tr> <td></td> <td> <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>DT_PLOTTER</td> <td>Vector plotter</td> </tr> <tr> <td>DT_RASDISPLAY</td> <td>Raster display</td> </tr> <tr> <td>DT_RASPRINTER</td> <td>Raster printer</td> </tr> <tr> <td>DT_RASCAMERA</td> <td>Raster camera</td> </tr> <tr> <td>DT_CHARSTREAM</td> <td>Character stream</td> </tr> <tr> <td>DT_METAFILE</td> <td>Metafile</td> </tr> <tr> <td>DT_DISPFIL</td> <td>Display file</td> </tr> </tbody> </table> </td> </tr> <tr> <td>HORZSIZE</td> <td>Width of the physical display, in millimeters.</td> </tr> <tr> <td>VERTSIZE</td> <td>Height of the physical display, in millimeters.</td> </tr> <tr> <td>HORZRES</td> <td>Width of the display, in pixels.</td> </tr> <tr> <td>VERTRES</td> <td>Height of the display, in raster lines.</td> </tr> <tr> <td>LOGPIXELSX</td> <td>Number of pixels per logical inch along the display width.</td> </tr> <tr> <td>LOGPIXELSY</td> <td>Number of pixels per logical inch along the display height.</td> </tr> <tr> <td>BITSPIXEL</td> <td>Number of adjacent color bits for each pixel.</td> </tr> <tr> <td>PLANES</td> <td>Number of color planes.</td> </tr> <tr> <td>NUMBRUSHES</td> <td>Number of device-specific brushes.</td> </tr> <tr> <td>NUMPENS</td> <td>Number of device-specific pens.</td> </tr> <tr> <td>NUMMARKERS</td> <td>Number of device-specific markers.</td> </tr> <tr> <td>NUMFONTS</td> <td>Number of device-specific fonts.</td> </tr> <tr> <td>NUMCOLORS</td> <td>Number of entries in the device's color table.</td> </tr> <tr> <td>ASPECTX</td> <td>Relative width of a device pixel used for line drawing.</td> </tr> <tr> <td>ASPECTY</td> <td>Relative height of a device pixel used for line drawing.</td> </tr> <tr> <td>ASPECTXY</td> <td>Diagonal width of a device pixel used for line drawing.</td> </tr> <tr> <td>PDEVICESIZE</td> <td>Size of the PDEVICE internal structure, in bytes.</td> </tr> <tr> <td>CLIPCAPS</td> <td>Clipping capabilities the device supports. It can be one of the following values:</td> </tr> <tr> <td></td> <td> <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>CP_NONE</td> <td>Output is not clipped.</td> </tr> <tr> <td>CP_RECTANGLE</td> <td>Output is clipped to rectangles.</td> </tr> <tr> <td>CP_REGION</td> <td>Output is clipped to regions.</td> </tr> </tbody> </table> </td> </tr> <tr> <td>SIZEPALETTE</td> <td>Number of entries in the system palette. This index is valid only if the device driver sets the RC_PALETTE bit in the RASTERCAPS index; it is available only if the driver is written for Windows 3.0 or later.</td> </tr> <tr> <td>NUMRESERVED</td> <td>Number of reserved entries in the system palette. This index is valid only if the device driver sets the RC_PALETTE bit in the RASTERCAPS index; it is</td> </tr> </tbody> </table>	Index	Description	DRIVERVERSION	Version number of the device driver.	TECHNOLOGY	Device technology. It can be one of the following values:		<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>DT_PLOTTER</td> <td>Vector plotter</td> </tr> <tr> <td>DT_RASDISPLAY</td> <td>Raster display</td> </tr> <tr> <td>DT_RASPRINTER</td> <td>Raster printer</td> </tr> <tr> <td>DT_RASCAMERA</td> <td>Raster camera</td> </tr> <tr> <td>DT_CHARSTREAM</td> <td>Character stream</td> </tr> <tr> <td>DT_METAFILE</td> <td>Metafile</td> </tr> <tr> <td>DT_DISPFIL</td> <td>Display file</td> </tr> </tbody> </table>	Value	Meaning	DT_PLOTTER	Vector plotter	DT_RASDISPLAY	Raster display	DT_RASPRINTER	Raster printer	DT_RASCAMERA	Raster camera	DT_CHARSTREAM	Character stream	DT_METAFILE	Metafile	DT_DISPFIL	Display file	HORZSIZE	Width of the physical display, in millimeters.	VERTSIZE	Height of the physical display, in millimeters.	HORZRES	Width of the display, in pixels.	VERTRES	Height of the display, in raster lines.	LOGPIXELSX	Number of pixels per logical inch along the display width.	LOGPIXELSY	Number of pixels per logical inch along the display height.	BITSPIXEL	Number of adjacent color bits for each pixel.	PLANES	Number of color planes.	NUMBRUSHES	Number of device-specific brushes.	NUMPENS	Number of device-specific pens.	NUMMARKERS	Number of device-specific markers.	NUMFONTS	Number of device-specific fonts.	NUMCOLORS	Number of entries in the device's color table.	ASPECTX	Relative width of a device pixel used for line drawing.	ASPECTY	Relative height of a device pixel used for line drawing.	ASPECTXY	Diagonal width of a device pixel used for line drawing.	PDEVICESIZE	Size of the PDEVICE internal structure, in bytes.	CLIPCAPS	Clipping capabilities the device supports. It can be one of the following values:		<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>CP_NONE</td> <td>Output is not clipped.</td> </tr> <tr> <td>CP_RECTANGLE</td> <td>Output is clipped to rectangles.</td> </tr> <tr> <td>CP_REGION</td> <td>Output is clipped to regions.</td> </tr> </tbody> </table>	Value	Meaning	CP_NONE	Output is not clipped.	CP_RECTANGLE	Output is clipped to rectangles.	CP_REGION	Output is clipped to regions.	SIZEPALETTE	Number of entries in the system palette. This index is valid only if the device driver sets the RC_PALETTE bit in the RASTERCAPS index; it is available only if the driver is written for Windows 3.0 or later.	NUMRESERVED	Number of reserved entries in the system palette. This index is valid only if the device driver sets the RC_PALETTE bit in the RASTERCAPS index; it is
Index	Description																																																																										
DRIVERVERSION	Version number of the device driver.																																																																										
TECHNOLOGY	Device technology. It can be one of the following values:																																																																										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>DT_PLOTTER</td> <td>Vector plotter</td> </tr> <tr> <td>DT_RASDISPLAY</td> <td>Raster display</td> </tr> <tr> <td>DT_RASPRINTER</td> <td>Raster printer</td> </tr> <tr> <td>DT_RASCAMERA</td> <td>Raster camera</td> </tr> <tr> <td>DT_CHARSTREAM</td> <td>Character stream</td> </tr> <tr> <td>DT_METAFILE</td> <td>Metafile</td> </tr> <tr> <td>DT_DISPFIL</td> <td>Display file</td> </tr> </tbody> </table>	Value	Meaning	DT_PLOTTER	Vector plotter	DT_RASDISPLAY	Raster display	DT_RASPRINTER	Raster printer	DT_RASCAMERA	Raster camera	DT_CHARSTREAM	Character stream	DT_METAFILE	Metafile	DT_DISPFIL	Display file																																																										
Value	Meaning																																																																										
DT_PLOTTER	Vector plotter																																																																										
DT_RASDISPLAY	Raster display																																																																										
DT_RASPRINTER	Raster printer																																																																										
DT_RASCAMERA	Raster camera																																																																										
DT_CHARSTREAM	Character stream																																																																										
DT_METAFILE	Metafile																																																																										
DT_DISPFIL	Display file																																																																										
HORZSIZE	Width of the physical display, in millimeters.																																																																										
VERTSIZE	Height of the physical display, in millimeters.																																																																										
HORZRES	Width of the display, in pixels.																																																																										
VERTRES	Height of the display, in raster lines.																																																																										
LOGPIXELSX	Number of pixels per logical inch along the display width.																																																																										
LOGPIXELSY	Number of pixels per logical inch along the display height.																																																																										
BITSPIXEL	Number of adjacent color bits for each pixel.																																																																										
PLANES	Number of color planes.																																																																										
NUMBRUSHES	Number of device-specific brushes.																																																																										
NUMPENS	Number of device-specific pens.																																																																										
NUMMARKERS	Number of device-specific markers.																																																																										
NUMFONTS	Number of device-specific fonts.																																																																										
NUMCOLORS	Number of entries in the device's color table.																																																																										
ASPECTX	Relative width of a device pixel used for line drawing.																																																																										
ASPECTY	Relative height of a device pixel used for line drawing.																																																																										
ASPECTXY	Diagonal width of a device pixel used for line drawing.																																																																										
PDEVICESIZE	Size of the PDEVICE internal structure, in bytes.																																																																										
CLIPCAPS	Clipping capabilities the device supports. It can be one of the following values:																																																																										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>CP_NONE</td> <td>Output is not clipped.</td> </tr> <tr> <td>CP_RECTANGLE</td> <td>Output is clipped to rectangles.</td> </tr> <tr> <td>CP_REGION</td> <td>Output is clipped to regions.</td> </tr> </tbody> </table>	Value	Meaning	CP_NONE	Output is not clipped.	CP_RECTANGLE	Output is clipped to rectangles.	CP_REGION	Output is clipped to regions.																																																																		
Value	Meaning																																																																										
CP_NONE	Output is not clipped.																																																																										
CP_RECTANGLE	Output is clipped to rectangles.																																																																										
CP_REGION	Output is clipped to regions.																																																																										
SIZEPALETTE	Number of entries in the system palette. This index is valid only if the device driver sets the RC_PALETTE bit in the RASTERCAPS index; it is available only if the driver is written for Windows 3.0 or later.																																																																										
NUMRESERVED	Number of reserved entries in the system palette. This index is valid only if the device driver sets the RC_PALETTE bit in the RASTERCAPS index; it is																																																																										



	available only if the driver is written for Windows 3.0 or later.
COLORRES	Color resolution of the device, in bits per pixel. This index is valid only if the device driver sets the RC_PALETTE bit in the RASTERCAPS index; it is available only if the driver is written for Windows 3.0 or later.
RASTERCAPS	Raster capabilities the device supports. It can be a combination of the following values:
	Value                      Meaning
	RC_BANDING                Supports banding.
	RC_BIGFONT               Supports fonts larger than 64K.
	RC_BITBLT                Transfers bitmaps.
	RC_BITMAP64             Supports bitmaps larger than 64K.
	RC_DEVBITS              Supports device bitmaps.
	RC_DI_BITMAP            Supports the <u>SetDIBits</u> and <u>GetDIBits</u> functions.
	RC_DIBTODEV            Supports the <u>SetDIBitsToDevice</u> function.
	RC_FLOODFILL           Performs flood fills.
	RC_GDI20_OUTPUT        Supports Windows version 2.0 features.
	RC_GDI20_STATE        Includes a state block in the device context.
	RC_NONE                 Supports no raster operations.
	RC_OP_DX_OUTPUT        Supports dev opaque and DX array.
	RC_PALETTE             Specifies a palette-based device.
	RC_SAVEBITMAP         Saves bitmaps locally.
	RC_SCALING             Supports scaling.
	RC_STRETCHBLT         Supports the <u>StretchBlt</u> function.
	RC_STRETCHDIB         Supports the <u>StretchDIBits</u> function.
CURVECAPS	Curve capabilities the device supports. It can be a combination of the following values:
	Value                      Meaning
	CC_NONE                 Supports curves.
	CC_CIRCLES              Supports circles.
	CC_PIE                   Supports pie wedges.
	CC_CHORD                Supports chords.
	CC_ELLIPSES            Supports ellipses.
	CC_WIDE                 Supports wide borders.
	CC_STYLED               Supports styled borders.
	CC_WIDESTYLED         Supports wide, styled borders.
	CC_INTERIORS           Supports interiors.
	CC_ROUNDRECT          Supports rectangles with rounded corners.
LINECAPS	Line capabilities the device supports. It can be a combination of the following values:
	Value                      Meaning
	LC_NONE                 Supports no lines.
	LC_POLYLINE            Supports polylines.

	LC_MARKER	Supports markers.
	LC_POLYMARKER	Supports polymarkers.
	LC_WIDE	Supports wide lines.
	LC_STYLED	Supports styled lines.
	LC_WIDESTYLED	Supports wide, styled lines.
	LC_INTERIORS	Supports interiors.
POLYGONALCAPS	Polygonal capabilities the device supports. It can be a combination of the following values:	
	Value	Meaning
	PC_NONE	Supports no polygons.
	PC_POLYGON	Supports alternate fill polygons.
	PC_RECTANGLE	Supports rectangles.
	PC_WINDPOLYGON	Supports winding number fill polygons.
	PC_SCANLINE	Supports scan lines.
	PC_WIDE	Supports wide borders.
	PC_STYLED	Supports styled borders.
	PC_WIDESTYLED	Supports wide, styled borders.
	PC_INTERIORS	Supports interiors.
TEXTCAPS	Text capabilities the device supports. It can be a combination of the following values:	
	Value	Meaning
	TC_OP_CHARACTER	Supports character output precision, which indicates the device can place device fonts at any pixel location. This is required for any device with device fonts.
	TC_OP_STROKE	Supports stroke output precision, which indicates the device can omit any stroke of a device font.
	TC_CP_STROKE	Supports stroke clip precision, which indicates the device can clip device fonts to a pixel boundary.
	TC_CR_90	Supports 90-degree character rotation, which indicates the device can rotate characters only 90 degrees at a time.
	TC_CR_ANY	Supports character rotation at any degree, which indicates the device can rotate device fonts through any angle.
	TC_SF_X_YINDEP	Supports scaling independent of x and y directions, which indicates the device can scale device fonts separately in x and y directions.
	TC_SA_DOUBLE	Supports doubled characters for scaling, which indicates the device can double the size of device fonts.

TC_SA_INTEGER	Supports integer multiples for scaling, which indicates the device can scale the size of device fonts in any integer multiple.
TC_SA_CONTIN	Supports any multiples for exact scaling, which indicates the device can scale device fonts by any amount but still preserve the x and y ratios.
TC_EA_DOUBLE	Supports double-weight characters, which indicates the device can make device fonts bold. If this bit is not set for printer drivers, graphics device interface (GDI) attempts to create bold device fonts by printing them twice.
TC_IA_ABLE	Supports italics, which indicates the device can make device fonts italic. If this bit is not set, GDI assumes italics are not available.
TC_UA_ABLE	Supports underlining, which indicates the device can underline device fonts. If this bit is not set, GDI creates underlines for device fonts.
TC_SO_ABLE	Supports strikeouts, which indicates the device can strikeout device fonts. If this bit is not set, GDI creates strikeouts for device fonts.
TC_RA_ABLE	Supports raster fonts, which indicates that GDI should enumerate any raster or TrueType fonts available for this device in response to a call to the EnumFonts or EnumFontFamilies function. If this bit is not set, GDI-supplied raster or TrueType fonts are not enumerated when these functions are called.
TC_VA_ABLE	Supports vector fonts, which indicates that GDI should enumerate any vector fonts available for this device in response to a call to the EnumFonts or EnumFontFamilies function. This is significant for vector devices only (that is, for plotters). Display drivers (which must be able to use raster

fonts) and raster printer drivers always enumerate vector fonts, because GDI rasterizes vector fonts before sending them to the driver.

TC\_RESERVED

Reserved; must be zero.

#### Returns

The return value is the value of the requested capability if the function is successful.

#### Example

The following example uses the `GetDeviceCaps` function to determine whether a device supports raster capabilities and is palette-based. If so, the example calls the `GetSystemPaletteUse` function.

**WORD** nUse;

```
hdc = GetDC(hwnd);
if ((GetDeviceCaps(hdc, RASTERCAPS) & RC_PALETTE) == 0) {
    ReleaseDC(hwnd, hdc);
    break;
}
nUse = GetSystemPaletteUse(hdc);
ReleaseDC(hwnd, hdc);
```

See Also

LOGFONT

### GetDIBits (3.0)

```
int GetDIBits(hdc, hbmp, nStartScan, cScanLines, lpvBits, lpbmi, fuColorUse)
HDC hdc; /* handle of device context */
HBITMAP hbmp;
* handle of bitmap
/
UINT nStartScan;
* first scan line to set in destination bitmap
/
UINT cScanLines;
* number of scan lines to copy
/
void FAR* lpvBits;
* address of array for bitmap bits
/
BITMAPINFO FAR* lpbmi;
* address of structure with bitmap data
/
UINT fuColorUse;
* type of color table
/
```

The GetDIBits function retrieves the bits of the specified bitmap and copies them, in device-independent format, into the buffer pointed to by the lpvBits parameter. The lpbmi parameter retrieves the color format for the device-independent bits.

Parameter	Description						
hdc	Identifies the device context.						
hbmp	Identifies the bitmap.						
nStartScan	Specifies the first scan line to be set in the bitmap received in the lpvBits parameter.						
cScanLines	Specifies the number of lines to be copied.						
lpvBits	Points to a buffer that will receive the bitmap bits in device-independent format.						
lpbmi	Points to a <u>BITMAPINFO</u> structure that specifies the color format and dimension for the device-independent bitmap.						
fuColorUse	Specifies whether the bmiColors members of the <u>BITMAPINFO</u> structure are to contain explicit RGB values or indices into the currently realized logical palette. The fuColorUse parameter must be one of the following values:						
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>DIB_PAL_COLORS</td><td>Color table is to consist of an array of 16-bit indices into the currently realized logical palette.</td></tr><tr><td>DIB_RGB_COLORS</td><td>Color table is to contain literal <u>RGB</u> values.</td></tr></tbody></table>	Value	Meaning	DIB_PAL_COLORS	Color table is to consist of an array of 16-bit indices into the currently realized logical palette.	DIB_RGB_COLORS	Color table is to contain literal <u>RGB</u> values.
Value	Meaning						
DIB_PAL_COLORS	Color table is to consist of an array of 16-bit indices into the currently realized logical palette.						
DIB_RGB_COLORS	Color table is to contain literal <u>RGB</u> values.						

#### Returns

The return value specifies the number of scan lines copied from the bitmap if the function is successful. Otherwise, it is zero.

#### Comments

If the lpvBits parameter is NULL, the GetDIBits function fills in the BITMAPINFO structure to which the lpbmi parameter points but does not retrieve bits from the bitmap.

The bitmap identified by the hbmp parameter must not be selected into a device context when the application calls this function.

The origin for device-independent bitmaps (DIBs) is the lower-left corner of the bitmap, not the upper-left corner, which is the origin when the mapping mode is MM\_TEXT.

#### See Also

SetDIBits, BITMAPINFO

### GetFontData (3.1)

```
DWORD GetFontData(hdc, dwTable, dwOffset, lpvBuffer, cbData)
HDC hdc; /* handle of device context */
DWORD dwTable; /* metric table to
query */
DWORD dwOffset;
/* offset into table being queried */
void FAR* lpvBuffer;
/* address of buffer for font data */
DWORD cbData;
/* length of data to query */
```

The GetFontData function retrieves font-metric information from a scalable font file. The information to retrieve is identified by specifying an offset into the font file and the length of the information to return.

Parameter	Description
hdc	Identifies the device context.
dwTable	Specifies the name of the metric table to be returned. This parameter can be one of the metric tables documented in the TrueType Font Files specification, published by Microsoft Corporation. If this parameter is zero, the information is retrieved starting at the beginning of the font file.
dwOffset	Specifies the offset from the beginning of the table at which to begin retrieving information. If this parameter is zero, the information is retrieved starting at the beginning of the table specified by the dwTable parameter. If this value is greater than or equal to the size of the table, GetFontData returns zero.
lpvBuffer	Points to a buffer that will receive the font information. If this value is NULL, the function returns the size of the buffer required for the font data specified in the dwTable parameter.
cbData	Specifies the length, in bytes, of the information to be retrieved. If this parameter is zero, GetFontData returns the size of the data specified in the dwTable parameter.

#### Returns

The return value specifies the number of bytes returned in the buffer pointed to by the lpvBuffer parameter, if the function is successful. Otherwise, it is -1.

#### Comments

An application can sometimes use the GetFontData function to save a TrueType font with a document. To do this, the application determines whether the font can be embedded and then retrieves the entire font file, specifying zero for the dwTable, dwOffset, and cbData parameters.

Applications can determine whether a font can be embedded by checking the otmfsType member of the `OUTLINETEXTMETRIC` structure. If bit 1 of otmfsType is set, embedding is not permitted for the font. If bit 1 is clear, the font can be embedded. If bit 2 is set, the embedding is read-only.

If an application attempts to use this function to retrieve information for a non-TrueType font, the GetFontData function returns -1.

#### Example

The following example retrieves an entire TrueType font file:

```
HGLOBAL hglb;
DWORD dwSize;
void FAR* lpvBuffer;

dwSize = GetFontData(hdc, NULL, 0L, NULL, 0L); /* get file size */

hglb = GlobalAlloc(GPTR, dwSize); /* allocate memory */
lpvBuffer = GlobalLock(hglb);
GetFontData(hdc, NULL, 0L, lpvBuffer, dwSize); /* retrieve data */
```

The following retrieves an entire TrueType font file 4K at a time:

```
#define BUFFER_SIZE 4096
BYTE Buffer[BUFFER_SIZE];
DWORD dwOffset;
DWORD dwSize;

dwOffset = 0L;
while(dwSize = GetFontData(hdc, NULL, dwOffset,
    Buffer, BUFFER_SIZE)) {
    . /* process data in buffer */
    .
    dwOffset += dwSize;
}
```

The following example retrieves a TrueType font table:

```
HGLOBAL hglb;
DWORD dwSize;
void FAR* lpvBuffer;

LPSTR lpszTable;
DWORD dwTable;

lpszTable = "cmap";
dwTable = *(LPDWORD) lpszTable; /* construct DWORD type */

dwSize = GetFontData(hdc, dwTable, 0L, NULL, 0L); /* get table size */

hglb = GlobalAlloc(GPTR, dwSize); /* allocate memory */
lpvBuffer = GlobalLock(hglb);
GetFontData(hdc, dwTable, 0L, lpvBuffer, dwSize); /* retrieve data */

See Also
GetOutlineTextMetrics, OUTLINETEXTMETRIC
```

### GetGlyphOutline (3.1)

```

DWORD GetGlyphOutline(hdc, uChar, fuFormat, lpgm, cbBuffer, lpBuffer, lpmat2)
HDC hdc; /* handle of device context */
UINT uChar; /* character to query */
/
UINT fuFormat; /* format of data to return */
/
LPGLYPHMETRICS lpgm; /* address of structure with glyph metrics */
/
DWORD cbBuffer; /* size of buffer for data */
/
void FAR* lpBuffer; /* address of buffer for outline data */
/
LPMAT2 lpmat2; /* address of structure with transform matrix */
/

```

The GetGlyphOutline function retrieves the outline curve or bitmap for an outline character in the current font.

Parameter	Description						
hdc	Identifies the device context.						
uChar	Specifies the character for which information is to be returned.						
fuFormat	Specifies the format in which the function is to return information. It can be one of the following values: <table border="1" data-bbox="446 1060 1395 1312"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td><u>GGO_BITMAP</u></td> <td>Returns the glyph bitmap. When the function returns, the buffer pointed to by the lpBuffer parameter contains a 1-bit-per-pixel bitmap whose rows start on doubleword boundaries.</td> </tr> <tr> <td><u>GGO_NATIVE</u></td> <td>Returns the curve data points in the rasterizer's native format, using device units. When this value is specified, any transformation specified in the lpmat2 parameter is ignored.</td> </tr> </tbody> </table> <p>When the value of this parameter is zero, the function fills in a <u>GLYPHMETRICS</u> structure but does not return glyph-outline data.</p>	Value	Meaning	<u>GGO_BITMAP</u>	Returns the glyph bitmap. When the function returns, the buffer pointed to by the lpBuffer parameter contains a 1-bit-per-pixel bitmap whose rows start on doubleword boundaries.	<u>GGO_NATIVE</u>	Returns the curve data points in the rasterizer's native format, using device units. When this value is specified, any transformation specified in the lpmat2 parameter is ignored.
Value	Meaning						
<u>GGO_BITMAP</u>	Returns the glyph bitmap. When the function returns, the buffer pointed to by the lpBuffer parameter contains a 1-bit-per-pixel bitmap whose rows start on doubleword boundaries.						
<u>GGO_NATIVE</u>	Returns the curve data points in the rasterizer's native format, using device units. When this value is specified, any transformation specified in the lpmat2 parameter is ignored.						
lpgm	Points to a <u>GLYPHMETRICS</u> structure that describes the placement of the glyph in the character cell.						
cbBuffer	Specifies the size of the buffer into which the function copies information about the outline character. If this value is zero and the fuFormat parameter is either the <u>GGO_BITMAP</u> or <u>GGO_NATIVE</u> values, the function returns the required size of the buffer.						
lpBuffer	Points to a buffer into which the function copies information about the outline character. If the fuFormat parameter specifies the <u>GGO_NATIVE</u> value, the information is copied in the form of <u>TTPOLYGONHEADER</u> and <u>TTPOLYCURVE</u> structures. If this value is NULL and the fuFormat parameter is either the <u>GGO_BITMAP</u> or <u>GGO_NATIVE</u> value, the function returns the required size of the buffer.						
lpmat2	Points to a <u>MAT2</u> structure that contains a transformation matrix for the character. This parameter cannot be NULL, even when the <u>GGO_NATIVE</u> value is specified for the fuFormat parameter.						

#### Returns

The return value is the size, in bytes, of the buffer required for the retrieved information if the cbBuffer parameter is zero or the lpBuffer parameter is NULL. Otherwise, it is a positive value if the function is successful, or -1 if there is an error.

#### Comments



An application can rotate characters retrieved in bitmap format by specifying a 2-by-2 transformation matrix in the structure pointed to by the `lpmat2` parameter.

A glyph outline is returned as a series of contours. Each contour is defined by a TPOLYGONHEADER structure followed by as many TPOLYCURVE structures as are required to describe it. All points are returned as POINTFX structures and represent absolute positions, not relative moves. The starting point given by the `pxStart` member of the TPOLYGONHEADER structure is the point at which the outline for a contour begins. The TPOLYCURVE structures that follow can be either polyline records or spline records. Polyline records are a series of points; lines drawn between the points describe the outline of the character. Spline records represent the quadratic curves used by TrueType (that is, quadratic b-splines).

For example, the `GetGlyphOutline` function retrieves the following information about the lowercase "i" in the Arial TrueType font:

```
dwrc = 88 /* total size of native buffer*/

TPOLYGONHEADER #1 /* contour for dot on i */
  cb= 44 /* size for contour */
  dwType = 24 /* TT_POLYGON_TYPE */
  pxStart = 1.000, 11.000

  TPOLYCURVE #1
    wType = TT_PRIM_LINE
    cpx = 3
    px[0] = 1.000, 12.000
    px[1] = 2.000, 12.000
    px[2] = 2.000, 11.000 /* automatically close to pxStart */

TPOLYGONHEADER #2 /* contour for body of i */
  cb= 44
  dwType = 24 /* TT_POLYGON_TYPE */
  pxStart = 1.000, 0.000

  TPOLYCURVE #1
    wType = TT_PRIM_LINE
    cpx = 3
    px[0] = 1.000, 9.000
    px[1] = 2.000, 9.000
    px[2] = 2.000, 0.000/* automatically close to pxStart */
```

See Also

[GetOutlineTextMetrics](#), [GLYPHMETRICS](#), [MAT2](#), [OUTLINETEXTMETRIC](#), [POINTFX](#), [TPOLYCURVE](#), [TPOLYGONHEADER](#)

## GGO\_BITMAP 1

Returns the glyph bitmap. When the function returns, the buffer pointed to by the lpBuffer parameter contains a 1-bit-per-pixel bitmap whose rows start on doubleword boundaries.

GGO\_BITMAP 1

## GGO\_NATIVE 2

Returns the curve data points in the rasterizer's native format, using device units. When this value is specified, any transformation specified in the `lpmat2` parameter is ignored.

GGO\_NATIVE 2

### GetKerningPairs (3.1)

```
int GetKerningPairs(hdc, cPairs, lpkrnpair)
HDC hdc; /* handle of device context */
int cPairs; /* number of kerning pairs */
/*
KERNINGPAIR FAR* lpkrnpair;
* pointer to structures for kerning pairs
*/
```

The GetKerningPairs function retrieves the character kerning pairs for the font that is currently selected in the specified device context.

Parameter	Description
hdc	Identifies a device context. The GetKerningPairs function retrieves kerning pairs for the current font for this device context.
cPairs	Specifies the number of <u>KERNINGPAIR</u> structures pointed to by the lpkrnpair parameter. The function will not copy more kerning pairs than specified by cPairs.
lpkrnpair	Points to an array of <u>KERNINGPAIR</u> structures that receive the kerning pairs when the function returns. This array must contain at least as many structures as specified by the cPairs parameter. If this parameter is NULL, the function returns the total number of kerning pairs for the font.

#### Returns

The return value specifies the number of kerning pairs retrieved or the total number of kerning pairs in the font, if the function is successful. It is zero if the function fails or there are no kerning pairs for the font.

#### See Also

[KERNINGPAIR](#)

## GetMapMode (2.x)

```
int GetMapMode(hdc)
HDC hdc;      /* handle of device context */
```

The GetMapMode function retrieves the current mapping mode.

Parameter	Description
hdc	Identifies the device context.

### Returns

The return value specifies the mapping mode if the function is successful.

It can be one of the following values:

Value	Meaning
MM_ANISOTROPIC	Logical units are converted to arbitrary units with arbitrarily scaled axes. Setting the mapping mode to MM_ANISOTROPIC does not change the current window or viewport settings. To change the units, orientation, and scaling, an application should use the <a href="#">SetWindowExt</a> and <a href="#">SetViewportExt</a> functions.
MM_HIENGLISH	Each logical unit is converted to 0.001 inch. Positive x is to the right; positive y is up.
MM_HIMETRIC	Each logical unit is converted to 0.01 millimeter. Positive x is to the right; positive y is up.
MM_ISOTROPIC	Logical units are converted to arbitrary units with equally scaled axes; that is, one unit along the x-axis is equal to one unit along the y-axis. The <a href="#">SetWindowExt</a> and <a href="#">SetViewportExt</a> functions must be used to specify the <u>desired units and the orientation of the axes</u> . GDI makes adjustments as necessary to ensure that the x and y units remain the same size.
MM_LOENGLISH	Each logical unit is converted to 0.01 inch. Positive x is to the right; positive y is up.
MM_LOMETRIC	Each logical unit is converted to 0.1 millimeter. Positive x is to the right; positive y is up.
MM_TEXT	Each logical unit is converted to one device pixel. Positive x is to the right; positive y is down.
MM_TWIPS	Each logical unit is converted to 1/20 of a point. (Because a point is 1/72 inch, a twip is 1/1440 inch). Positive x is to the right; positive y is up.

### Example

The following example uses the GetMapMode function to determine whether the current mapping mode is MM\_TEXT:

```
if (GetMapMode(hdc) != MM_TEXT) {
    TextOut(hdc, 100, -200, "Mapping mode must be MM_TEXT", 28);
    return FALSE;
}
```

### See Also

[SetMapMode](#)

## GetMetaFile (2.x)

```
HMETAFILE GetMetaFile(lpszFile)
LPCSTR lpszFile;      /* address of metafile name */
```

The GetMetaFile function creates a handle of a specified metafile.

Parameter	Description
lpszFile	Points to the null-terminated string that specifies the MS-DOS filename of the metafile. The metafile is assumed to exist.

### Returns

The return value is the handle of a metafile if the function is successful. Otherwise, it is NULL.

### Example

The following example uses the [CopyMetaFile](#) function to copy a metafile to a specified file, plays the copied metafile, uses the [GetMetaFile](#) function to retrieve a handle to the copied metafile, uses the [SetWindowOrg](#) function to change the position at which the metafile is played 200 logical units to the right, and then plays the metafile at the new location:

```
HANDLE hmf, hmfSource, hmfOld;
LPSTR lpszFile1 = "MFTTest";

hmf = CopyMetaFile(hmfSource, lpszFile1);
PlayMetaFile(hdc, hmf);
DeleteMetaFile(hmf);

hmfOld = GetMetaFile(lpszFile1);
SetWindowOrg(hdc, -200, 0);
PlayMetaFile(hdc, hmfOld);

DeleteMetaFile(hmfSource);
DeleteMetaFile(hmfOld);
```

See Also  
[CopyMetaFile](#), [PlayMetaFile](#), [SetWindowOrg](#)



## GetMetaFileBits (2.x)

```
HGLOBAL GetMetaFileBits(hmf)
HMETAFILE hmf;          /* handle of metafile */
```

The GetMetaFileBits function returns a handle of the global memory object that contains the specified metafile as a collection of bits. The memory object can be used to determine the size of the metafile or to save the metafile as a file. The memory object should not be modified.

Parameter	Description
hmf	Identifies the memory metafile.

### Returns

The return value is the handle of the global memory object that contains the metafile, if the function is successful. Otherwise, it is NULL.

### Comments

The handle contained in the hmf parameter becomes invalid when the GetMetaFileBits function returns, so the returned global memory handle must be used to refer to the metafile.

When it no longer requires a global memory object that is associated with a metafile, an application should remove the object by using the [GlobalFree](#) function.

### See Also

[GlobalFree](#)

## GetNearestColor (2.x)

```
COLORREF GetNearestColor(hdc, clrref)
HDC hdc; /* handle of device context */
COLORREF clrref; /* color to match */
```

The GetNearestColor function retrieves the solid color that best matches a specified logical color; the given device must be able to represent this solid color.

Parameter	Description
hdc	Identifies the device context.
clrref	Specifies the color to be matched.

### Returns

The return value specifies an RGB (red, green, blue) color value that defines the solid color closest to the clrref value that the device can represent.

### See Also

[GetNearestPaletteIndex](#)

## GetNearestPaletteIndex (3.0)

```
UINT GetNearestPaletteIndex(hpal, clrref)
HPALETTE hpal;          /* handle of palette */
COLORREF clrref;      /* color to match */
```

The `GetNearestPaletteIndex` function retrieves the index of the logical-palette entry that best matches the specified color value.

Parameter	Description
<code>hpal</code>	Identifies the logical palette.
<code>clrref</code>	Specifies the color to be matched.

### Returns

The return value is the index of the logical-palette entry whose corresponding color best matches the specified color.

### Example

The following example uses the `GetNearestPaletteIndex` function to retrieve a color index from a palette. It then creates a brush with that retrieved color by using the `PALETTEINDEX` macro in a call to the `CreateSolidBrush` function.

```
WORD nColor;
HPALETTE hpal;
DWORD dwBrushColors[8][8];
HBRUSH hbr;
int x, y;

    . /* Initialize the array of brush colors. */
    .
nColor = GetNearestPaletteIndex(hpal, dwBrushColors[x][y]);
hbr = CreateSolidBrush(PALETTEINDEX(nColor));

    . /* Use the brush handle. */
    .
DeleteObject(hbr);
```

See Also

[CreateSolidBrush](#), [GetNearestColor](#), [GetPaletteEntries](#), [GetSystemPaletteEntries](#), [PALETTEINDEX](#)

## GetObject (2.x)

```
int GetObject(hgdiobj, cbBuffer, lpvObject)
HGDIOBJ hgdiobj;          /* handle of object          */
int cbBuffer;             /* size
of buffer for object information
/
void FAR* lpvObject;
* address of buffer for object information
/
```

The `GetObject` function fills a buffer with information that defines a given object. The function retrieves a `LOGPEN`, `LOGBRUSH`, `LOGFONT`, or `BITMAP` structure, or an integer, depending on the specified object.

Parameter	Description
<code>hgdiobj</code>	Identifies a logical pen, brush, font, bitmap, or palette.
<code>cbBuffer</code>	Specifies the number of bytes to be copied to the buffer.
<code>lpvObject</code>	Points to the buffer that is to receive the information.

### Returns

The return value specifies the number of bytes retrieved if the function is successful. Otherwise, it is zero.

### Comments

The buffer pointed to by the `lpvObject` parameter must be sufficiently large to receive the information.

If the `hgdiobj` parameter identifies a bitmap, the `GetObject` function returns only the width, height, and color format information of the bitmap. The bits can be retrieved by using the `GetBitmapBits` function.

If `hgdiobj` identifies a logical palette, `GetObject` retrieves an integer that specifies the number of entries in the palette; the function does not retrieve the `LOGPALETTE` structure that defines the palette. To retrieve information about palette entries, an application can call the `GetPaletteEntries` function.

### Example

The following example uses the `GetObject` function to fill a `LOGBRUSH` structure with the attributes of the current brush and then tests whether the brush style is `BS_SOLID`:

```
LOGBRUSH lb;

HBRUSH hbr;

GetObject(hbr, sizeof(LOGBRUSH), (LPSTR) &lb);
if (lb.lbStyle == BS_SOLID) {
    .
    .
    .
}
```

### See Also

`GetBitmapBits`, `GetPaletteEntries`, `GetStockObject`, `BITMAP`, `LOGBRUSH`, `LOGFONT`, `LOGPALETTE`, `LOGPEN`

### GetOutlineTextMetrics (3.1)

```
WORD GetOutlineTextMetrics(hdc, cbData, lpotm)
HDC hdc; /* handle of device context */
UINT cbData; /* size of buffer for information */
OUTLINETEXTMETRIC FAR* lpotm; /* address of structure for metrics */
```

The GetOutlineTextMetrics function retrieves metric information for TrueType fonts.

Parameter	Description
hdc	Identifies the device context.
cbData	Specifies the size, in bytes, of the buffer to which information is returned.
lpotm	Points to an <u>OUTLINETEXTMETRIC</u> structure. If this parameter is NULL, the function returns the size of the buffer required for the retrieved metric information.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The OUTLINETEXTMETRIC structure contains most of the font metric information provided with the TrueType format, including a TEXTMETRIC structure. The last four members of the OUTLINETEXTMETRIC structure are pointers to strings. Applications should allocate space for these strings in addition to the space required for the other members. Because there is no system-imposed limit to the size of the strings, the simplest method for allocating memory is to retrieve the required size by specifying NULL for the lpotm parameter in the first call to the GetOutlineTextMetrics function.

#### See Also

GetTextMetrics, OUTLINETEXTMETRIC, TEXTMETRIC

### GetPaletteEntries (3.0)

```
UINT GetPaletteEntries(hpal, iStart, cEntries, lppe)
HPALETTE hpal; /* handle of palette */
UINT iStart; /* first palette entry to retrieve */
/* number of entries to retrieve */
PALETTEENTRY FAR* lppe; /* address of structure for palette entries */
```

The GetPaletteEntries function retrieves a range of palette entries in a logical palette.

Parameter	Description
hpal	Identifies the logical palette.
iStart	Specifies the first logical-palette entry to be retrieved.
cEntries	Specifies the number of logical-palette entries to be retrieved.
lppe	Points to an array of <u>PALETTEENTRY</u> structures that will receive the palette entries. The array must contain at least as many structures as specified by the cEntries parameter.

#### Returns

The return value is the number of entries retrieved from the logical palette, if the function is successful. Otherwise, it is zero.

#### See Also

GetSystemPaletteEntries, PALETTEENTRY

## GetPixel (2.x)

```
COLORREF GetPixel(hdc, nXPos, nYPos)
HDC hdc;          /* handle of device context */
int nXPos;        /* x-coordinate of pixel to retrieve */
int nYPos;        /* y-coordinate of pixel to retrieve */
```

The `GetPixel` function retrieves the RGB (red, green, blue) color value of the pixel at the specified coordinates. The point must be in the clipping region; if it is not, the function is ignored.

Parameter	Description
<code>hdc</code>	Identifies the device context.
<code>nXPos</code>	Specifies the logical x-coordinate of the point to be examined.
<code>nYPos</code>	Specifies the logical y-coordinate of the point to be examined.

### Returns

The return value specifies an RGB color value for the color of the given point, if the function is successful. It is -1 if the coordinates do not specify a point in the clipping region.

### Comments

Not all devices support the `GetPixel` function.

### See Also

[GetDeviceCaps](#), [SetPixel](#)

## GetPolyFillMode (2.x)

```
int GetPolyFillMode(hdc)
HDC hdc;      /* handle of device context */
```

The GetPolyFillMode function retrieves the current polygon-filling mode.

Parameter	Description
hdc	Identifies the device context.

### Returns

The return value specifies the polygon-filling mode, ALTERNATE or WINDING, if the function is successful.

### Comments

When the polygon-filling mode is ALTERNATE, the system fills the area between odd-numbered and even-numbered polygon sides on each scan line. That is, the system fills the area between the first and second side, between the third and fourth side, and so on.

When the polygon-filling mode is WINDING, the system uses the direction in which a figure was drawn to determine whether to fill an area. Each line segment in a polygon is drawn in either a clockwise or a counterclockwise direction. Whenever an imaginary line drawn from an enclosed area to the outside of a figure passes through a clockwise line segment, a count is incremented. When the line passes through a counterclockwise line segment, the count is decremented. The area is filled if the count is nonzero when the line reaches the outside of the figure.

### Example

The following example uses the GetPolyFillMode function to determine whether the current polygon-filling mode is ALTERNATE:

```
int nPolyFillMode;

nPolyFillMode = GetPolyFillMode(hdc);
if (nPolyFillMode == ALTERNATE) {
    .
    .
    .
}
```

### See Also

[SetPolyFillMode](#)



### GetRasterizerCaps (3.1)

```
BOOL GetRasterizerCaps(lpraststat, cb)
RASTERIZER_STATUS FAR* lpraststat;          /* address of structure for status */
int cb;                                     /* number of bytes in structure */
/
```

The GetRasterizerCaps function returns flags indicating whether TrueType fonts are installed in the system.

Parameter	Description
lpraststat	Points to a <u>RASTERIZER_STATUS</u> structure that receives information about the rasterizer.
cb	Specifies the number of bytes that will be copied into the structure pointed to by the lpraststat parameter.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The GetRasterizerCaps function enables applications and printer drivers to determine whether TrueType is installed.

If the TT\_AVAILABLE flag is set in the wFlags member of the RASTERIZER\_STATUS structure, at least one TrueType font is installed. If the TT\_ENABLED flag is set, TrueType is enabled for the system.

#### See Also

GetOutlineTextMetrics, RASTERIZER\_STATUS

### GetRgnBox (3.0)

```
int GetRgnBox(hrgn, lprc)
HRGN hrgn;          /* handle of region          */
RECT FAR* lprc;     /* address of
structure with rectangle
/
```

The GetRgnBox function retrieves the coordinates of the bounding rectangle of the given region.

Parameter	Description
hrgn	Identifies the region.
lprc	Points to a <u>RECT</u> structure that receives the coordinates of the bounding rectangle.

#### Returns

The return value is SIMPLEREGION (region has no overlapping borders), COMPLEXREGION (region has overlapping borders), or NULLREGION (region is empty), if the function is successful. Otherwise, the return value is ERROR.

#### Example

The following example uses the GetRgnBox function to determine the type of a region:

```
RECT rc;
HRGN hrgn;
int RgnType;

RgnType = GetRgnBox(hrgn, &rc);

if (RgnType == COMPLEXREGION)
    TextOut(hdc, 10, 10, "COMPLEXREGION", 13);
else if (RgnType == SIMPLEREGION)
    TextOut(hdc, 10, 10, "SIMPLEREGION", 12);
else
    TextOut(hdc, 10, 10, "NULLREGION", 10);
```

#### See Also

RECT

## GetROP2 (2.x)

```
int GetROP2(hdc)
HDC hdc; /* handle of device context */
```

The GetROP2 function retrieves the current drawing mode. The drawing mode specifies how the colors of the pen and the interior of filled objects are combined with the color already on the screen surface.

Parameter	Description
hdc	Identifies the device context.

### Returns

The return value specifies the drawing mode if the function is successful.

### Comments

The drawing mode is for raster devices only and does not apply to vector devices. It can be any of the following values:

Value	Meaning
R2_BLACK	Pixel is always black.
R2_WHITE	Pixel is always white.
R2_NOP	Pixel remains unchanged.
R2_NOT	Pixel is the inverse of the screen color.
R2_COPYPEN	Pixel is the pen color.
R2_NOTCOPYPEN	Pixel is the inverse of the pen color.
R2_MERGEPEENNOT	Pixel is a combination of the pen color and the inverse of the screen color (final pixel = (~screen pixel)   pen).
R2_MASKPENNOT	Pixel is a combination of the colors common to both the pen and the inverse of the screen (final pixel = (~screen pixel) & pen).
R2_MERGENOTPEN	Pixel is a combination of the screen color and the inverse of the pen color (final pixel = (~pen)   screen pixel).
R2_MASKNOTPEN	Pixel is a combination of the colors common to both the screen and the inverse of the pen (final pixel = (~pen) & screen pixel).
R2_MERGEPEN	Pixel is a combination of the pen color and the screen color (final pixel = pen   screen pixel).
R2_NOTMERGEPEN	Pixel is the inverse of the R2_MERGEPEN color (final pixel = ~(pen   screen pixel)).
R2_MASKPEN	Pixel is a combination of the colors common to both the pen and the screen (final pixel = pen & screen pixel).
R2_NOTMASKPEN	Pixel is the inverse of the R2_MASKPEN color (final pixel = ~(pen & screen pixel)).
R2_XORPEN	Pixel is a combination of the colors that are in the pen and in the screen, but not in both (final pixel = pen ^ screen pixel).
R2_NOTXORPEN	Pixel is the inverse of the R2_XORPEN color (final pixel = ~(pen ^ screen pixel)).

### Example

The following example uses the GetROP2 function to test whether the current drawing mode is R2\_COPYPEN:

```
int nROP;

nROP = GetROP2(hdc);
if (nROP == R2_COPYPEN)
    TextOut(hdc, 100, 100, "ROP is R2_COPYPEN.", 18);
```

### See Also

[GetDeviceCaps](#), [SetROP2](#)

## GetStockObject (2.x)

```
HGDIOBJ GetStockObject(fnObject)
int fnObject; /* type of stock object */
```

The GetStockObject function retrieves a handle of one of the predefined stock pens, brushes, or fonts.

Parameter	Description																																				
fnObject	Specifies the type of stock object for which to retrieve a handle. This parameter can be one of the following values:																																				
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>BLACK_BRUSH</u></td><td>Black brush.</td></tr><tr><td><u>DKGRAY_BRUSH</u></td><td>Dark-gray brush.</td></tr><tr><td><u>GRAY_BRUSH</u></td><td>Gray brush.</td></tr><tr><td><u>HOLLOW_BRUSH</u></td><td>Hollow brush.</td></tr><tr><td><u>LTGRAY_BRUSH</u></td><td>Light-gray brush.</td></tr><tr><td><u>NULL_BRUSH</u></td><td>Null brush.</td></tr><tr><td><u>WHITE_BRUSH</u></td><td>White brush.</td></tr><tr><td><u>BLACK_PEN</u></td><td>Black pen.</td></tr><tr><td><u>NULL_PEN</u></td><td>Null pen.</td></tr><tr><td><u>WHITE_PEN</u></td><td>White pen.</td></tr><tr><td><u>ANSI_FIXED_FONT</u></td><td>Windows fixed-pitch system font.</td></tr><tr><td><u>ANSI_VAR_FONT</u></td><td>Windows variable-pitch system font.</td></tr><tr><td><u>DEVICE_DEFAULT_FONT</u></td><td>Device-dependent font.</td></tr><tr><td><u>OEM_FIXED_FONT</u></td><td>OEM-dependent fixed font.</td></tr><tr><td><u>SYSTEM_FONT</u></td><td>System font. By default, Windows uses the system font to draw menus, dialog box controls, and other text. In Windows versions 3.0 and later, the system font is a variable-pitch font width; earlier versions of Windows use a fixed-pitch system font.</td></tr><tr><td><u>SYSTEM_FIXED_FONT</u></td><td>Fixed-pitch system font used in Windows versions earlier than 3.0. This object is available for compatibility with earlier versions of Windows.</td></tr><tr><td><u>DEFAULT_PALETTE</u></td><td>Default color palette. This palette consists of the static colors in the system palette.</td></tr></tbody></table>	Value	Meaning	<u>BLACK_BRUSH</u>	Black brush.	<u>DKGRAY_BRUSH</u>	Dark-gray brush.	<u>GRAY_BRUSH</u>	Gray brush.	<u>HOLLOW_BRUSH</u>	Hollow brush.	<u>LTGRAY_BRUSH</u>	Light-gray brush.	<u>NULL_BRUSH</u>	Null brush.	<u>WHITE_BRUSH</u>	White brush.	<u>BLACK_PEN</u>	Black pen.	<u>NULL_PEN</u>	Null pen.	<u>WHITE_PEN</u>	White pen.	<u>ANSI_FIXED_FONT</u>	Windows fixed-pitch system font.	<u>ANSI_VAR_FONT</u>	Windows variable-pitch system font.	<u>DEVICE_DEFAULT_FONT</u>	Device-dependent font.	<u>OEM_FIXED_FONT</u>	OEM-dependent fixed font.	<u>SYSTEM_FONT</u>	System font. By default, Windows uses the system font to draw menus, dialog box controls, and other text. In Windows versions 3.0 and later, the system font is a variable-pitch font width; earlier versions of Windows use a fixed-pitch system font.	<u>SYSTEM_FIXED_FONT</u>	Fixed-pitch system font used in Windows versions earlier than 3.0. This object is available for compatibility with earlier versions of Windows.	<u>DEFAULT_PALETTE</u>	Default color palette. This palette consists of the static colors in the system palette.
Value	Meaning																																				
<u>BLACK_BRUSH</u>	Black brush.																																				
<u>DKGRAY_BRUSH</u>	Dark-gray brush.																																				
<u>GRAY_BRUSH</u>	Gray brush.																																				
<u>HOLLOW_BRUSH</u>	Hollow brush.																																				
<u>LTGRAY_BRUSH</u>	Light-gray brush.																																				
<u>NULL_BRUSH</u>	Null brush.																																				
<u>WHITE_BRUSH</u>	White brush.																																				
<u>BLACK_PEN</u>	Black pen.																																				
<u>NULL_PEN</u>	Null pen.																																				
<u>WHITE_PEN</u>	White pen.																																				
<u>ANSI_FIXED_FONT</u>	Windows fixed-pitch system font.																																				
<u>ANSI_VAR_FONT</u>	Windows variable-pitch system font.																																				
<u>DEVICE_DEFAULT_FONT</u>	Device-dependent font.																																				
<u>OEM_FIXED_FONT</u>	OEM-dependent fixed font.																																				
<u>SYSTEM_FONT</u>	System font. By default, Windows uses the system font to draw menus, dialog box controls, and other text. In Windows versions 3.0 and later, the system font is a variable-pitch font width; earlier versions of Windows use a fixed-pitch system font.																																				
<u>SYSTEM_FIXED_FONT</u>	Fixed-pitch system font used in Windows versions earlier than 3.0. This object is available for compatibility with earlier versions of Windows.																																				
<u>DEFAULT_PALETTE</u>	Default color palette. This palette consists of the static colors in the system palette.																																				

### Returns

The return value is the handle of the specified object if the function is successful. Otherwise, it is NULL.

### Comments

The DKGRAY\_BRUSH, GRAY\_BRUSH, and LTGRAY\_BRUSH objects should be used only in windows with the CS\_HREDRAW and CS\_VREDRAW class styles. Using a gray stock brush in any other style of window can lead to misalignment of brush patterns after a window is moved or sized. The origins of stock brushes cannot be adjusted.

### Example

The following example retrieves the handle of a black brush by calling the GetStockObject function, selects the brush into the device context, and fills a rectangle by using the black brush:

```
HBRUSH hbr, hbrOld;

hbr = GetStockObject(BLACK_BRUSH);
hbrOld = SelectObject(hdc, hbr);
Rectangle(hdc, 10, 10, 100, 100);
```

### See Also

GetObject, SetBrushOrg

BLACK\_BRUSH 4

Black brush.

BLACK\_BRUSH 4

DKGRAY\_BRUSH 3

Dark-gray brush.

DKGRAY\_BRUSH 3



GRAY\_BRUSH 2

Gray brush.

GRAY\_BRUSH 2

HOLLOW\_BRUSH NULL\_BRUSH

Hollow brush.

HOLLOW\_BRUSH NULL\_BRUSH

LTGRAY\_BRUSH 1

Light-gray brush.

LTGRAY\_BRUSH 1

NULL\_BRUSH 5

Null brush.

NULL\_BRUSH 5



WHITE\_BRUSH 0

White brush.

WHITE\_BRUSH 0

BLACK\_PEN 7

Black pen.

BLACK\_PEN 7

NULL\_PEN 8

Null pen.

NULL\_PEN 8

WHITE\_PEN 6

White pen.

WHITE\_PEN 6



ANSI\_FIXED\_FONT 11

Windows fixed-pitch system font.

ANSI\_FIXED\_FONT 11

ANSI\_VAR\_FONT 12

Windows variable-pitch system font.

ANSI\_VAR\_FONT 12

DEVICE\_DEFAULT\_FONT 14

Device-dependent font.

DEVICE\_DEFAULT\_FONT 14

OEM\_FIXED\_FONT 10  
OEM-dependent fixed font.

OEM\_FIXED\_FONT 10



## SYSTEM\_FONT 13

System font. By default, Windows uses the system font to draw menus, dialog box controls, and other text. In Windows versions 3.0 and later, the system font is a variable-pitch font width; earlier versions of Windows use a fixed-pitch system font.

SYSTEM\_FONT 13

**SYSTEM\_FIXED\_FONT 16**

Fixed-pitch system font used in Windows versions earlier than 3.0. This object is available for compatibility with earlier versions of Windows.

SYSTEM\_FIXED\_FONT 16

DEFAULT\_PALETTE 15

Default color palette. This palette consists of the static colors in the system palette.

DEFAULT\_PALETTE 15

## GetStretchBltMode (2.x)

```
int GetStretchBltMode(hdc)
HDC hdc;      /* handle of device context */
```

The GetStretchBltMode function retrieves the current bitmap-stretching mode. The bitmap-stretching mode defines how information is removed from bitmaps that were compressed by using the [StretchBlt](#) function.

Parameter	Description
hdc	Identifies the device context.

### Returns

The return value specifies the current bitmap-stretching mode--[STRETCH\\_ANDSCANS](#), [STRETCH\\_DELETESCANS](#), or [STRETCH\\_ORSCANS](#)--if the function is successful.

### Comments

The [STRETCH\\_ANDSCANS](#) and [STRETCH\\_ORSCANS](#) modes are typically used to preserve foreground pixels in monochrome bitmaps. The [STRETCH\\_DELETESCANS](#) mode is typically used to preserve color in color bitmaps.

### Example

The following example uses the GetStretchBltMode function to determine whether the current bitmap-stretching mode is [STRETCH\\_DELETESCANS](#); if so, it uses the [StretchBlt](#) function to display a compressed bitmap.

```
HDC hdcMem;

int nStretchMode;

nStretchMode = GetStretchBltMode(hdc);
if (nStretchMode == STRETCH\_DELETESCANS) {
    StretchBlt(hdc, 50, 175, 32, 32, hdcMem, 0, 0, 64, 64,
SRCCOPY);
    .
    .
    .
}
```

### See Also

[SetStretchBltMode](#), [StretchBlt](#)

### GetSystemPaletteEntries (3.0)

```
UINT GetSystemPaletteEntries(hdc, iStart, cEntries, lppe)
HDC hdc; /* handle of device context */
UINT iStart;
/* first palette entry to retrieve
/
UINT cEntries;
/* number of entries to retrieve
/
PALETTEENTRY FAR* lppe;
/* address of structure for palette entries
/
```

The GetSystemPaletteEntries function retrieves a range of palette entries from the system palette.

Parameter	Description
hdc	Identifies the device context.
iStart	Specifies the first system-palette entry to be retrieved.
cEntries	Specifies the number of system-palette entries to be retrieved.
lppe	Points to an array of <u>PALETTEENTRY</u> structures that receives the palette entries. The array must contain at least as many structures as specified by the cEntries parameter.

#### Returns

The return value is the number of entries retrieved from the system palette, if the function is successful. Otherwise, it is zero.

#### Example

The following example uses the GetDeviceCaps function to determine whether the specified device is palette-based. If the device supports palettes, the GetSystemPaletteEntries function is called, using GetDeviceCaps again, this time to determine the number of entries in the system palette.

```
PALETTEENTRY pe[MAXNUMBER];

hdc = GetDC(hwnd);
if (!(GetDeviceCaps(hdc, RASTERCAPS) & RC_PALETTE)) {
    ReleaseDC(hwnd, hdc);
    break;
}
GetSystemPaletteEntries(hdc, 0, GetDeviceCaps(hdc, SIZEPALETTE),
    pe);
ReleaseDC(hwnd, hdc);
```

#### See Also

GetDeviceCaps, GetPaletteEntries, PALETTEENTRY



## GetSystemPaletteUse (3.0)

```
UINT GetSystemPaletteUse(hdc)
HDC hdc; /* handle of device context */
```

The `GetSystemPaletteUse` function determines whether an application has access to the entire system palette.

Parameter	Description
<code>hdc</code>	Identifies the device context. This device context must support color palettes.

### Returns

The return value specifies the current use of the system palette, if the function is successful. This parameter can be one of the following values:

Value	Meaning
<code><u>SYSPAL_NOSTATIC</u></code>	System palette contains no static colors except black and white.
<code><u>SYSPAL_STATIC</u></code>	System palette contains static colors that do not change when an application realizes its logical palette.

### Comments

The system palette contains 20 default static colors that are not changed when an application realizes its logical palette. An application can gain access to most of these colors by calling the `SetSystemPaletteUse` function.

### Example

The following example uses the `GetDeviceCaps` function to determine whether the specified device is palette-based. If the device supports palettes, the `GetSystemPaletteUse` function is called.

```
WORD nUse;
```

```
hdc = GetDC(hwnd);
if ((GetDeviceCaps(hdc, RASTERCAPS) & RC_PALETTE) == 0) {
    ReleaseDC(hwnd, hdc);
    break;
}
nUse = GetSystemPaletteUse(hdc);
ReleaseDC(hwnd, hdc);
```

### See Also

`GetDeviceCaps`, `SetSystemPaletteUse`

SYSPAL\_NOSTATIC 2

System palette contains no static colors except black and white.

SYSPAL\_NOSTATIC 2

`SYSPAL_STATIC` 1

System palette contains static colors that do not change when an application realizes its logical palette.

SYSPAL\_STATIC 1

## GetTextCharacterExtra (2.x)

```
int GetTextCharacterExtra(hdc)
HDC hdc;      /* handle of device context */
```

The `GetTextCharacterExtra` function retrieves the current setting for the amount of intercharacter spacing. Graphics device interface (GDI) adds this spacing to each character, including break characters, when it writes a line of text to the device context.

Parameter	Description
<code>hdc</code>	Identifies the device context.

### Returns

The return value specifies the amount of intercharacter spacing if the function is successful.

### Comments

The default value for the amount of intercharacter spacing is zero.

### See Also

[SetTextCharacterExtra](#)

## GetTextAlign (2.x)

```
UINT GetTextAlign(hdc)
HDC hdc; /* handle of device context */
```

The GetTextAlign function retrieves the status of the text-alignment flags for the given device context.

Parameter	Description
hdc	Identifies the device context.

### Returns

The return value specifies the status of the text-alignment flags. This parameter can be one or more of the following values:

Value	Meaning
<u>TA_BASELINE</u>	Specifies alignment of the x-axis and the base line of the chosen font within the bounding rectangle.
<u>TA_BOTTOM</u>	Specifies alignment of the x-axis and the bottom of the bounding rectangle.
<u>TA_CENTER</u>	Specifies alignment of the y-axis and the center of the bounding rectangle.
<u>TA_LEFT</u>	Specifies alignment of the y-axis and the left side of the bounding rectangle.
<u>TA_NOUPDATECP</u>	Specifies that the current position is not updated.
<u>TA_RIGHT</u>	Specifies alignment of the y-axis and the right side of the bounding rectangle.
<u>TA_TOP</u>	Specifies alignment of the x-axis and the top of the bounding rectangle.
<u>TA_UPDATECP</u>	Specifies that the current position is updated.

### Comments

The text-alignment flags retrieved by the GetTextAlign function are used by the TextOut and ExtTextOut functions. These flags determine how TextOut and ExtTextOut align a string of text in relation to the string's starting point.

The text-alignment flags are not necessarily single-bit flags and may be equal to zero. To test whether a flag is set, an application should follow three steps:

- 1 Apply the bitwise OR operator to the flag and its related flags.

Following are the groups of related flags:

- TA\_LEFT, TA\_CENTER, and TA\_RIGHT
- TA\_BASELINE, TA\_BOTTOM, and TA\_TOP
- TA\_NOUPDATECP and TA\_UPDATECP

- 2 Apply the bitwise AND operator to the result and the return value of the GetTextAlign function.

- 3 Test for the equality of this result and the flag.

### Example

The following example uses the method described in the preceding Comments section to determine whether text is aligned at the right, left, or center of the bounding rectangle. If the TA\_RIGHT flag is set, the SetTextAlign function is used to set the text alignment to the left side of the rectangle.

```
switch ((TA_LEFT | TA_CENTER | TA_RIGHT) & GetTextAlign(hdc)) {
    case TA_RIGHT:
        TextOut(hdc, 200, 100, "This is TA_RIGHT.", 17);
        SetTextAlign(hdc, TA_LEFT);
        TextOut(hdc, 200, 120, "This is TA_LEFT.", 16);
        break;
    case TA_LEFT:
        .
        .
        .
    case TA_CENTER:
        .
        .
        .
}
```

### See Also

ExtTextOut, SetTextAlign, TextOut

TA\_BASELINE 0x0018

Specifies alignment of the x-axis and the base line of the chosen font within the bounding rectangle.



TA\_BASELINE 0x0018

TA\_BOTTOM 0x0008

Specifies alignment of the x-axis and the bottom of the bounding rectangle.

TA\_BOTTOM 0x0008

TA\_CENTER 0x0006

Specifies alignment of the y-axis and the center of the bounding rectangle.

TA\_CENTER 0x0006

TA\_LEFT 0x0000

Specifies alignment of the y-axis and the left side of the bounding rectangle.

TA\_LEFT 0x0000

TA\_NOUPDATECP 0x0000

Specifies that the current position is not updated.



TA\_NOUPDATECP 0x0000

TA\_RIGHT 0x0002

Specifies alignment of the y-axis and the right side of the bounding rectangle.

TA\_RIGHT 0x0002

TA\_TOP 0x0000

Specifies alignment of the x-axis and the top of the bounding rectangle.

TA\_TOP 0x0000

TA\_UPDATECP 0x0001

Specifies that the current position is updated.

TA\_UPDATECP 0x0001

## GetTextColor (2.x)

```
COLORREF GetTextColor(hdc)
HDC hdc;      /* handle of device context */
```

The GetTextColor function retrieves the current text color. The text color is the foreground color of characters drawn by using the graphics device interface (GDI) text-output functions.

Parameter	Description
hdc	Identifies the device context.

### Returns

The return value specifies the current text color as a red, green, blue (RGB) color value, if the function is successful.

### Example

The following example sets the text color to red if the GetTextColor function determines that the current text color is black:

```
DWORD dwColor;

dwColor = GetTextColor(hdc);
if (dwColor == RGB(0, 0, 0)) /* if current color is black */
    SetTextColor(hdc, RGB(255, 0, 0)); /* sets color to red */
```

### See Also

GetBkColor, GetBkMode, SetBkMode, SetTextColor, RGB



## GetTextExtent (2.x)

```
DWORD GetTextExtent(hdc, lpszString, cbString)
HDC hdc; /* handle of device context */
LPCSTR lpszString; /* address of
string
/
int cbString;
* number of bytes in string
/
```

The GetTextExtent function computes the width and height of a line of text, using the current font to compute the dimensions.

Parameter	Description
hdc	Identifies the device context.
lpszString	Points to a character string.
cbString	Specifies the number of bytes in the string.

### Returns

The low-order word of the return value contains the string width, in logical units, if the function is successful; the high-order word contains the string height.

### Comments

The current clipping region does not affect the width and height returned by the GetTextExtent function.

Since some devices do not place characters in regular cell arrays (that is, they kern characters), the sum of the extents of the characters in a string may not be equal to the extent of the string.

### Example

The following example retrieves the number of characters in a string by using the `lstrlen` function, calls the `GetTextExtent` function to retrieve the dimensions of the string, and then uses the `LOWORD` macro to determine the string width, in logical units:

```
DWORD dwExtent;
WORD wTextWidth;
LPSTR lpszJustified = "Text to be justified in this test.";

dwExtent = GetTextExtent(hdc, lpszJustified, lstrlen(lpszJustified));
wTextWidth = LOWORD(dwExtent);
```

### See Also

GetTabbedTextExtent, SetTextJustification

### GetTextExtentPoint (3.1)

```
BOOL GetTextExtentPoint(hdc, lpszString, cbString, lpSize)
HDC hdc;          /* handle of device context */
LPCSTR lpszString; /* address of text string */
int cbString;     /* number of bytes in string */
SIZE FAR* lpSize; /* address if structure for string size */
```

The GetTextExtentPoint function computes the width and height of the specified text string. The GetTextExtentPoint function uses the currently selected font to compute the dimensions of the string. The width and height, in logical units, are computed without considering any clipping.

The GetTextExtentPoint function may be used as either a wide-character function (where text arguments must use Unicode) or an ANSI function (where text arguments must use characters from the Windows 3.x character set).

Parameter	Description
hdc	Identifies the device context.
lpszString	Points to a text string.
cbString	Specifies the number of bytes in the text string.
lpSize	Points to a <u>SIZE</u> structure that will receive the dimensions of the string.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

Because some devices do not place characters in regular cell arrays--that is, because they carry out kerning--the sum of the extents of the characters in a string may not be equal to the extent of the string.

The calculated width takes into account the intercharacter spacing set by the SetTextCharacterExtra function.

#### See Also

SetTextCharacterExtra

## GetTextFace (2.x)

```
int GetTextFace(hdc, cbBuffer, lpszFace)
HDC hdc;          /* handle of device context */
int cbBuffer;     /* size of buffer for face */
LPCTSTR lpszFace;
/* pointer to buffer for face name */
```

The `GetTextFace` function copies the typeface name of the current font into a buffer. The typeface name is copied as a null-terminated string.

Parameter	Description
<code>hdc</code>	Identifies the device context.
<code>cbBuffer</code>	Specifies the buffer size, in bytes. If the typeface name is longer than the number of bytes specified by this parameter, the name is truncated.
<code>lpszFace</code>	Points to the buffer for the typeface name.

### Returns

The return value specifies the number of bytes copied to the buffer, not including the terminating null character, if the function is successful. Otherwise, it is zero.

### Example

The following example uses the `GetTextFace` function to retrieve the name of the current typeface, calls the `SetTextAlign` function so that the current position is updated when the `TextOut` function is called, and then writes some introductory text and the name of the typeface by calling `TextOut`:

```
int nFaceNameLen;
char aFaceName[80];

nFaceNameLen = GetTextFace(hdc, /* returns length of string */
    sizeof(aFaceName), /* size of face-name buffer */
    (LPCTSTR) aFaceName); /* address of face-name buffer */

SetTextAlign(hdc,
    TA_UPDATECP); /* updates current position */
MoveTo(hdc, 100, 100); /* sets current position*/
TextOut(hdc, 0, 0, /* uses current position for text */
    "This is the current face name: ", 31);
TextOut(hdc, 0, 0, aFaceName, nFaceNameLen);
```

### See Also

GetTextMetrics, SetTextAlign, TextOut

## GetTextMetrics (2.x)

```
BOOL GetTextMetrics(hdc, lptm)
HDC hdc;          /* handle of device context */
TEXTMETRIC FAR* lptm;
/* pointer to structure for font metrics
/
```

The GetTextMetrics function retrieves the metrics for the current font.

Parameter	Description
hdc	Identifies the device context.
lptm	Points to the <u>TEXTMETRIC</u> structure that receives the metrics.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Example

The following example calls the GetTextMetrics function and then uses information in a TEXTMETRIC structure to determine how many break characters are in a string of text:

```
TEXTMETRIC tm;
int j, cBreakChars, cchString;
LPSTR lpszJustified = "Text to be justified in this test.";

GetTextMetrics(hdc, &tm);

cchString = lstrlen(lpszJustified);

for (cBreakChars = 0, j = 0; j < cchString; j++)
    if(*(lpszJustified + j) == (char) tm.tmBreakChar)
        cBreakChars++;
```

### See Also

GetTextAlign, GetTextExtent, GetTextFace, SetTextJustification, TEXTMETRIC

## GetViewportExt (2.x)

```
DWORD GetViewportExt(hdc)
HDC hdc; /* handle of device context */
```

The GetViewportExt function retrieves the x- and y-extents of the device context's viewport.

Parameter	Description
hdc	Identifies the device context.

### Returns

The low-order word of the return value contains the x-extent, in device units, if the function is successful; the high-order word contains the y-extent.

### Example

The following example uses the GetViewportExt function and the LOWORD and HIWORD macros to retrieve the x- and y-extents for a device context:

```
HDC hdc;
DWORD dw;
int xViewExt, yViewExt;

hdc = GetDC(hwnd);
dw = GetViewportExt(hdc);
ReleaseDC(hwnd, hdc);
xViewExt = LOWORD(dw);
yViewExt = HIWORD(dw);
```

### See Also

GetViewportExtEx, SetViewportExt

### GetViewportExtEx (3.1)

BOOL GetViewportExtEx(hdc, lpSize)  
HDC hdc;  
SIZE FAR\* lpSize;

The GetViewportExtEx function retrieves the x- and y-extents of the device context's viewport.

Parameter	Description
hdc	Identifies the device context.
lpSize	Points to a <u>SIZE</u> structure. The x- and y-extents (in device units) are placed in this structure.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### See Also

GetViewportExt, SetViewportExt, SetViewportExtEx

## GetViewportOrg (2.x)

```
DWORD GetViewportOrg(hdc)
HDC hdc; /* handle of device context */
```

The `GetViewportOrg` function retrieves the x- and y-coordinates of the origin of the viewport associated with the given device context.

Parameter	Description
<code>hdc</code>	Identifies the device context.

### Returns

The low-order word of the return value contains the viewport origin's x-coordinate, in device coordinates, if the function is successful; the high-order word contains the y-coordinate of the viewport origin.

### Example

The following example uses the `GetViewportOrg` function and the `LOWORD` and `HIWORD` macros to retrieve the x- and y-coordinates of the viewport origin:

```
HDC hdc;
DWORD dw;
int xViewOrg, yViewOrg;

hdc = GetDC(hwnd);
dw = GetViewportOrg(hdc);
ReleaseDC(hwnd, hdc);
xViewOrg = LOWORD(dw);
yViewOrg = HIWORD(dw);
```

### See Also

[GetViewportOrgEx](#), [GetWindowOrg](#), [SetViewportOrg](#)

## GetViewportOrgEx (3.1)

BOOL GetViewportOrgEx(hdc, lpPoint)

HDC hdc;

POINT FAR\* lpPoint;

The GetViewportOrgEx function retrieves the x- and y-coordinates of the origin of the viewport associated with the specified device context.

Parameter	Description
hdc	Identifies the device context.
lpPoint	Points to a <u>POINT</u> structure. The origin of the viewport (in device coordinates) is placed in this structure.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### See Also

GetViewportOrg, SetViewportOrg, SetViewportOrgEx



## GetWindowExt (2.x)

```
DWORD GetWindowExt(hdc)
HDC hdc; /* handle of device context */
```

The GetWindowExt function retrieves the x- and y-extents of the window associated with the given device context.

Parameter	Description
hdc	Identifies the device context.

### Returns

The return value specifies the x- and y-extents, in logical units, if the function is successful. The x-extent is in the low-order word; the y-extent is in the high-order word.

### Example

The following example uses the GetWindowExt function and the LOWORD and HIWORD macros to retrieve the x- and y-extents of a window:

```
HDC hdc;
DWORD dw;
int xWindExt, yWindExt;

hdc = GetDC(hwnd);
dw = GetWindowExt(hdc);
ReleaseDC(hwnd, hdc);
xWindExt = LOWORD(dw);
yWindExt = HIWORD(dw);
```

### See Also

GetWindowExtEx, SetWindowExt

### GetWindowExtEx (3.1)

BOOL GetWindowExtEx(hdc, lpSize)  
HDC hdc;  
SIZE FAR\* lpSize;

This function retrieves the x- and y-extents of the window associated with the specified device context.

Parameter	Description
hdc	Identifies the device context.
lpSize	Points to a <u>SIZE</u> structure. The x- and y-extents (in logical units) are placed in this structure.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### See Also

GetWindowExt, SetWindowExt, SetWindowExtEx

## GetWindowOrg (2.x)

```
DWORD GetWindowOrg(hdc)
HDC hdc;      /* handle of device context */
```

The GetWindowOrg function retrieves the x- and y-coordinates of the origin of the window associated with the given device context.

Parameter	Description
hdc	Identifies the device context.

### Returns

The low-order word of the return value contains the logical x-coordinate of the window's origin, if the function is successful; the high-order word contains the y-coordinate.

### Example

The following example uses the GetWindowOrg function and the LOWORD and HIWORD macros to retrieve the x- and y-coordinates for the window origin:

```
HDC hdc;
DWORD dw;
int xWindOrg, yWindOrg;

hdc = GetDC(hwnd);
dw = GetWindowOrg(hdc);
ReleaseDC(hwnd, hdc);
xWindOrg = LOWORD(dw);
yWindOrg = HIWORD(dw);
```

### See Also

GetViewportOrg, GetWindowOrgEx, SetWindowOrg

## GetWindowOrgEx (3.1)

BOOL GetWindowOrgEx(hdc, lpPoint)

HDC hdc;

POINT FAR\* lpPoint;

The GetWindowOrgEx function retrieves the x- and y-coordinates of the origin of the window associated with the specified device context.

Parameter	Description
hdc	Identifies the device context.
lpPoint	Points to a <u>POINT</u> structure. The origin of the window (in logical coordinates) is placed in this structure.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### See Also

GetWindowOrg, SetWindowOrg, SetWindowOrgEx

## IntersectClipRect (2.x)

```
int IntersectClipRect(hdc, nLeftRect, nTopRect, nRightRect, nBottomRect)
HDC hdc;          /* handle of device context */
int nLeftRect;   /* x-
coordinate top-left corner of rectangle
/
int nTopRect;
* y-coordinate top-left corner of rectangle
/
int nRightRect;
* x-coordinate bottom-right corner of rectangle
/
int nBottomRect;
* y-coordinate bottom-right corner of rectangle
/
```

The `IntersectClipRect` function creates a new clipping region from the intersection of the current region and a specified rectangle.

Parameter	Description
<code>hdc</code>	Identifies the device context.
<code>nLeftRect</code>	Specifies the logical x-coordinate of the upper-left corner of the rectangle.
<code>nTopRect</code>	Specifies the logical y-coordinate of the upper-left corner of the rectangle.
<code>nRightRect</code>	Specifies the logical x-coordinate of the lower-right corner of the rectangle.
<code>nBottomRect</code>	Specifies the logical y-coordinate of the lower-right corner of the rectangle.

### Returns

The return value specifies that the resulting region has overlapping borders (`COMPLEXREGION`), is empty (`NULLREGION`), or has no overlapping borders (`SIMPLEREGION`). Otherwise, the return value is `ERROR`.

### Comments

An application uses the `IntersectClipRect` function to create a clipping region from the intersection of the current region and a specified rectangle. An application can also create a clipping region that is the intersection of two regions, by specifying `RGN_AND` in a call to the `CombineRgn` function and then making this combined region the clipping region by calling the `SelectClipRgn` function.

The width of the rectangle, specified by the absolute value of `nRightRect - nLeftRect`, must not exceed 32,767 units. This limit applies to the height of the rectangle as well.

### Example

The following example creates a square clipping region and colors it red by using a red brush to fill the client area. The `IntersectClipRect` function is called with coordinates that overlap the region, and the client area is filled with a yellow brush. The only region colored yellow is the overlap between the region and the coordinates specified in the call to `IntersectClipRect`.

```
RECT rc;
HREGION hrgn;
HBRUSH hbrRed, hbrYellow;

GetClientRect(hwnd, &rc);
hrgn = CreateRectRgn(10, 10, 110, 110);
SelectClipRgn(hdc, hrgn);
hbrRed = CreateSolidBrush(RGB(255, 0, 0));
FillRect(hdc, &rc, hbrRed);

IntersectClipRect(hdc, 100, 100, 200, 200);

hbrYellow = CreateSolidBrush(RGB(255, 255, 0));
FillRect(hdc, &rc, hbrYellow);

DeleteObject(hbrRed);
DeleteObject(hbrYellow);
```

DeleteObject(hrgn) ;

See Also

CombineRgn, SelectClipRgn

## InvertRgn (2.x)

```
BOOL InvertRgn(hdc, hrgn)
HDC hdc;          /* handle of device context */
HRGN hrgn;        /* handle of region */
```

The InvertRgn function inverts the colors in a given region.

Parameter	Description
hdc	Identifies the device context.
hrgn	Identifies the region for which colors are to be inverted.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

On monochrome screens, the InvertRgn function makes white pixels black and black pixels white. On color screens, the inversion depends on how the colors are generated for the screen.

### Example

The following example sets the device coordinates of and creates a rectangular region, selects the region into a device context, and then calls the InvertRgn function to display the region in inverted colors:

```
HRGN hrgn;

hrgn = CreateRectRgn(10, 10, 110, 110);
SelectObject(hdc, hrgn);
InvertRgn(hdc, hrgn);
```

```
DeleteObject(hrgn);
```

### See Also

[FillRgn](#), [PaintRgn](#)

## IsGDIObject (3.1)

```
BOOL IsGDIObject(hobj)
HGDIOBJ hobj;    /* handle of a menu */
```

The IsGDIObject function determines whether the specified handle is not the handle of a graphics device interface (GDI) object.

Parameter	Description
hobj	Specifies a handle to test.

### Returns

The return value is nonzero if the handle may be the handle of a GDI object. It is zero if the handle is not the handle of a GDI object.

### Comments

An application cannot use IsGDIObject to guarantee that a given handle is to a GDI object. However, this function can be used to guarantee that a given handle is not to a GDI object.

### See Also

GetObject



## LineDDA (2.x)

```
void LineDDA(nXStart, nYStart, nXEnd, nYEnd, lnddaprc, lParam)
int nXStart;          /* x-coordinate of line beginning */
int nYStart;          /* y-
coordinate of line beginning
/
int nXEnd;
* x-coordinate of line end
/
int nYEnd;
* y-coordinate of line end
/
LINEDDAPROC lnddaprc;
* address of callback function
/
LPARAM lParam;
* address of application-defined data
/
```

The LineDDA function computes all successive points in a line specified by starting and ending coordinates. For each point on the line, the system calls an application-defined callback function, specifying the coordinates of that point.

Parameter	Description
nXStart	Specifies the logical x-coordinate of the first point.
nYStart	Specifies the logical y-coordinate of the first point.
nXEnd	Specifies the logical x-coordinate of the endpoint. This endpoint is not part of the line.
nYEnd	Specifies the logical y-coordinate of the endpoint. This endpoint is not part of the line.
lnddaprc	Specifies the procedure-instance address of the application-defined callback function. The address must have been created by using the <a href="#">MakeProcInstance</a> function. For more information about the callback function, see the description of the <a href="#">LineDDAProc</a> callback function.
lParam	Points to 32 bits of application-defined data that is passed to the callback function.

### Returns

This function does not return a value.

### Example

The following example uses the LineDDA function to draw a dot every two spaces between the beginning and ending points of a line:

```
/* Callback function */

void CALLBACK DrawDots(int xPos, int yPos, LPSTR lphdc)
{
    static short cSpaces = 1;

    if (cSpaces == 3) {

        /* Draw a black dot. */

        SetPixel(* (HDC FAR*) lphdc, xPos, yPos, 0);

        /* Initialize the space count. */

        cSpaces = 1;
    }
    else
        cSpaces++;
}
```

See Also

LineDDAProc, MakeProcInstance

## LineTo (2.x)

```
BOOL LineTo(hdc, xEnd, yEnd)
HDC hdc;      /* handle of device context      */
int xEnd;     /* x-coordinate of line endpoint */
int yEnd;     /* y-coordinate of line endpoint */
/
```

The LineTo function draws a line from the current position up to, but not including, the specified endpoint. The function uses the selected pen to draw the line and sets the current position to the coordinates (xEnd, yEnd).

Parameter	Description
hdc	Identifies the device context.
xEnd	Specifies the logical x-coordinate of the line's endpoint.
yEnd	Specifies the logical y-coordinate of the line's endpoint.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Example

The following example sets the current position by using the MoveTo function before calling the LineTo function. The example uses POINT structures to store the coordinates.

```
HDC hdc;
```

```
POINT ptStart = { 12, 12 };
POINT ptEnd = { 128, 135 };
```

```
MoveTo(hdc, ptStart.x, ptStart.y);
LineTo(hdc, ptEnd.x, ptEnd.y);
```

See Also

MoveTo, POINT

## LPtoDP (2.x)

```
BOOL LPtoDP(hdc, lppt, cPoints)
HDC hdc;          /* handle of device context */
POINT FAR* lppt; /* address of
array with points
/
int cPoints;
* number of points in array
/
```

The LPtoDP function converts logical coordinates (points) into device coordinates.

Parameter	Description
hdc	Identifies the device context.
lppt	Points to an array of <u>POINT</u> structures. The coordinates in each structure are mapped to the device coordinates of the current device context.
cPoints	Specifies the number of points in the array.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The conversion depends on the current mapping mode and the settings of the origins and extents of the device's window and viewport.

The x- and y-coordinates of points are 2-byte signed integers in the range -32,768 through 32,767. In cases where the mapping mode would result in values larger than these limits, the system sets the values to -32,768 and 32,767, respectively.

### Example

The following example sets the mapping mode to MM\_LOENGLISH and then calls the LPtoDP function to convert the coordinates of a rectangle into device coordinates:

```
RECT rc;
```

```
SetMapMode(hdc, MM_LOENGLISH);
```

```
SetRect(&rc, 100, -100, 200, -200);
```

```
LPtoDP(hdc, (LPPOINT) &rc, 2);
```

See Also

DPtoLP, POINT

## MoveTo (2.x)

```
DWORD MoveTo(hdc, x, y)
HDC hdc;      /* handle of device context      */
int x;        /* x-coordinate of new position */
int y;        /* y-coordinate of new position */
/
```

The MoveTo function moves the current position to the specified coordinates.

Parameter	Description
hdc	Identifies the device context.
x	Specifies the logical x-coordinate of the new position.
y	Specifies the logical y-coordinate of the new position.

### Returns

The low-order word of the return value contains the logical x-coordinate of the previous position, if the function is successful; the high-order word contains the logical y-coordinate.

### Example

The following example uses the MoveTo function to set the current position and then calls the LineTo function. The example uses POINT structures to store the coordinates.

```
HDC hdc;
```

```
POINT ptStart = { 12, 12 };
POINT ptEnd = { 128, 135 };
```

```
MoveTo(hdc, ptStart.x, ptStart.y);
```

```
LineTo(hdc, ptEnd.x, ptEnd.y);
```

See Also

GetCurrentPosition, LineTo, POINT

### MoveToEx (3.1)

```
BOOL MoveToEx(hdc, x, y, lpPoint)
HDC hdc;          /* handle of device context */
int x;            /* x-coordinate
of new position */
int y;            /* y-coordinate of new position */
POINT FAR* lpPoint;
/* pointer to structure for previous position */
```

The MoveToEx function moves the current position to the point specified by the x and y parameters, optionally returning the previous position.

Parameter	Description
hdc	Identifies the device context.
x	Specifies the logical x-coordinate of the new position.
y	Specifies the logical y-coordinate of the new position.
lpPoint	Points to a POINT structure in which the previous current position will be stored. If this parameter is <u>NULL</u> , no previous position is returned.

#### Returns

The return value is nonzero if the call is successful. Otherwise, it is zero.

#### See Also

MoveTo, POINT

## OffsetClipRgn (2.x)

```
int OffsetClipRgn(hdc, nXOffset, nYOffset)
HDC hdc;          /* device-context handle */
int nXOffset;    /* offset along x-axis */
int nYOffset;
* offset along y-axis
/
```

The OffsetClipRgn function moves the clipping region of the given device by the specified offsets.

Parameter	Description
hdc	Identifies the device context.
nXOffset	Specifies the number of logical units to move left or right.
nYOffset	Specifies the number of logical units to move up or down.

### Returns

The return value is SIMPLEREGION (region has no overlapping borders), COMPLEXREGION (region has overlapping borders), or NULLREGION (region is empty), if the function is successful. Otherwise, the return value is ERROR.

### Example

The following example creates an elliptical region and selects it as the clipping region for a device context. The OffsetClipRgn function is called repeatedly to move the clipping region from left to right across the screen. Because only the new clipping region is redrawn each time the Rectangle function is called, the left side of each ellipse remains on the screen when the clipping region moves. When the loop has finished, a wide blue line with rounded ends stretches from one side of the client area to the other.

```
RECT rc;
HRGN hrgn;
HBRUSH hbr, hbrPrevious;
int i;

GetClientRect(hwnd, &rc);
hrgn = CreateEllipticRgn(0, 100, 100, 200);
SelectClipRgn(hdc, hrgn);
hbr = CreateSolidBrush(RGB(0, 0, 255));
hbrPrevious = SelectObject(hdc, hbr);

for (i = 0; i < rc.right - 100; i++) {
    OffsetClipRgn(hdc, 1, 0);
    Rectangle(hdc, rc.left, rc.top, rc.right, rc.bottom);
}

SelectObject(hdc, hbrPrevious);
DeleteObject(hbr);
DeleteObject(hrgn);
```

### See Also

CreateEllipticRgn, SelectClipRgn

## OffsetRgn (2.x)

```
int OffsetRgn(hrgn, nXOffset, nYOffset)
HRGN hrgn; /* handle of region */
int nXOffset; /* offset along x-axis */
int nYOffset; /* offset along y-axis */
/
```

The OffsetRgn function moves the given region by the specified offsets.

Parameter	Description
hrgn	Identifies the region to be moved.
nXOffset	Specifies the number of logical units to move left or right.
nYOffset	Specifies the number of logical units to move up or down.

### Returns

The return value is SIMPLEREGION (region has no overlapping borders), COMPLEXREGION (region has overlapping borders), or NULLREGION (region is empty), if the function is successful. Otherwise, the return value is ERROR.

### Comments

The coordinate values of a region must not be greater than 32,767 or less than -32,768. The nXOffset and nYOffset parameters must be carefully chosen to prevent invalid regions.

### Example

The following example creates a rectangular region, uses the OffsetRgn function to move the region 50 positive units in the x- and y-directions, selects the offset region into the device context, and then fills it by using a blue brush:

```
HDC hdcLocal;
HRGN hrgn;
HBRUSH hbrBlue;
int RgnType;

hdcLocal = GetDC(hwnd);
hrgn = CreateRectRgn(100, 10, 210, 110);
SelectObject(hdc, hrgn);
PaintRgn(hdc, hrgn);

RgnType = OffsetRgn(hrgn, 50, 50);
SelectObject(hdc, hrgn);

if (RgnType == ERROR)
    TextOut(hdcLocal, 10, 135, "ERROR", 5);
else if (RgnType == SIMPLEREGION)
    TextOut(hdcLocal, 10, 135, "SIMPLEREGION", 12);
else if (RgnType == NULLREGION)
    TextOut(hdcLocal, 10, 135, "NULLREGION", 10);
else
    TextOut(hdcLocal, 10, 135, "Unrecognized value.", 19);

hbrBlue = CreateSolidBrush(RGB(0, 0, 255));
FillRgn(hdc, hrgn, hbrBlue);

DeleteObject(hrgn);
DeleteObject(hbrBlue);
ReleaseDC(hwnd, hdcLocal);
```



## OffsetViewportOrg (2.x)

DWORD OffsetViewportOrg(hdc, nXOffset, nYOffset)

HDC hdc; /\* handle of device context \*/

int nXOffset;

/\* offset along x-axis \*/

int nYOffset;

\* offset along y-axis

/

The OffsetViewportOrg function modifies the coordinates of the viewport origin relative to the coordinates of the current viewport origin.

Parameter	Description
hdc	Identifies the device context.
nXOffset	Specifies the value, in device units, to add to the x-coordinate of the current origin.
nYOffset	Specifies the value, in device units, to add to the y-coordinate of the current origin.

### Returns

The low-order word of the return value contains the x-coordinate, in device units, of the previous viewport origin, if the function is successful; the high-order word contains the y-coordinate.

### Comments

The viewport origin is the origin of the device coordinate system for a window. By changing the viewport origin, an application can change the way the graphics device interface (GDI) maps points from the logical coordinate system. GDI maps all points in the logical coordinate system to the viewport in the same way as it maps the origin.

To map points to the right, specify a negative value for the nXOffset parameter. Similarly, to map points down (in the MM\_TEXT mapping mode), specify a negative value for the nYOffset parameter.

### Example

The following example uses the OffsetWindowOrg and OffsetViewportOrg functions to reposition the output of the PlayMetaFile function on the screen:

```
HDC hdcMeta;
```

```
HANDLE hmf;
```

```
hdcMeta = CreateMetaFile((LPSTR) NULL);
```

```
    . /* Record the metafile. */
```

```
    .
```

```
PlayMetaFile(hdc, hmf);
```

```
OffsetWindowOrg(hdc, -200, -200);
```

```
PlayMetaFile(hdc, hmf); /* MM_TEXT screen output +200 x, +200 y */
```

```
OffsetViewportOrg(hdc, 0, -200);
```

```
PlayMetaFile(hdc, hmf); /* outputs -200 y from last PlayMetaFile */
```

```
DeleteMetaFile(hmf);
```

See Also

GetViewportOrg, OffsetWindowOrg, SetViewportOrg

### OffsetViewportOrgEx (3.1)

```
BOOL OffsetViewportOrgEx(hdc, nX, nY, lpPoint)
HDC hdc; /* handle of device context */ /* device units to
int nX; add to x-coordinate
/
int nY; * device units to add to y-coordinate
/
POINT FAR* lpPoint;
* address of POINT structure
/
```

The OffsetViewportOrgEx function modifies the viewport origin relative to the current values. The formulas are written as follows:

$$\begin{aligned}x_{\text{NewVO}} &= x_{\text{OldVO}} + X \\y_{\text{NewVO}} &= y_{\text{OldVO}} + Y\end{aligned}$$

The new origin is the sum of the current origin and the nX and nY values.

Parameter	Description
hdc	Identifies the device context.
nX	Specifies the number of device units to add to the current origin's x-coordinate.
nY	Specifies the number of device units to add to the current origin's y-coordinate.
lpPoint	Points to a <u>POINT</u> structure. The previous viewport origin (in device coordinates) is placed in this structure. If lpPoint is NULL, the previous viewport origin is not returned.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

## OffsetWindowOrg (2.x)

DWORD OffsetWindowOrg(hdc, nXOffset, nYOffset)

HDC hdc; /\* handle of device context \*/

int nXOffset; /\* offset along x-axis \*/

int nYOffset;

\* offset along y-axis

/

The OffsetWindowOrg function modifies the window origin relative to the coordinates of the current window origin.

Parameter	Description
hdc	Identifies the device context.
nXOffset	Specifies the value, in logical units, to add to the x-coordinate of the current origin.
nYOffset	Specifies the value, in logical units, to add to y-coordinate of the current origin.

### Returns

The low-order word of the return value contains the logical x-coordinate of the previous window origin, if the function is successful; the high-order word contains the logical y-coordinate.

### Comments

The window origin is the origin of the logical coordinate system for a window. By changing the window origin, an application can change the way the graphics device interface (GDI) maps logical points to the physical coordinate system (the viewport). GDI maps all points in the logical coordinate system to the viewport in the same way as it maps the origin.

To map points to the right, specify a negative value for the nXOffset parameter. Similarly, to map points down (in the MM\_TEXT mapping mode), specify a negative value for the nYOffset parameter.

### Example

The following example uses the OffsetWindowOrg and OffsetViewportOrg functions to reposition the output of the PlayMetaFile function on the screen:

```
HDC hdcMeta;
```

```
HANDLE hmf;
```

```
hdcMeta = CreateMetaFile((LPSTR) NULL);
```

```
    . /* Record the metafile. */
```

```
    .
```

```
PlayMetaFile(hdc, hmf);
```

```
OffsetWindowOrg(hdc, -200, -200);
```

```
PlayMetaFile(hdc, hmf); /* MM_TEXT screen output +200 x, +200 y */
```

```
OffsetViewportOrg(hdc, 0, -200);
```

```
PlayMetaFile(hdc, hmf); /* outputs -200 y from last PlayMetaFile */
```

```
DeleteMetaFile(hmf);
```

See Also

GetWindowOrg, OffsetViewportOrg, SetWindowOrg

### OffsetWindowOrgEx (3.1)

```
BOOL OffsetWindowOrgEx(hdc, nX, nY, lpPoint)
HDC hdc;                /* handle of device context */
int nX;                 /* logical units
to add to x-coordinate */
int nY;                 /* logical units
to add to y-coordinate */
POINT FAR* lpPoint;
/* address of POINT structure */
```

The OffsetWindowOrgEx function modifies the viewport origin relative to the current values. The formulas are written as follows:

$$\begin{aligned}x_{\text{NewWO}} &= x_{\text{OldWO}} + X \\ y_{\text{NewWO}} &= y_{\text{OldWO}} + Y\end{aligned}$$

The new origin is the sum of the current origin and the nX and nY values.

Parameter	Description
hdc	Identifies the device context.
nX	Specifies the number of logical units to add to the current origin's x-coordinate.
nY	Specifies the number of logical units to add to the current origin's y-coordinate.
lpPoint	Points to a <u>POINT</u> structure. The previous window origin (in logical coordinates) is placed in this structure. If lpPoint is NULL, the previous origin is not returned.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

## PaintRgn (2.x)

```
BOOL PaintRgn(hdc, hrgn)
HDC hdc;          /* handle of device context */
HRGN hrgn;        /* handle of region */
```

The PaintRgn function fills a region by using the current brush for the given device context.

Parameter	Description
hdc	Identifies the device context that contains the region to be filled.
hrgn	Identifies the region to be filled. The coordinates for the given region are specified in device units.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Example

The following example uses the current brush for a device context to fill an elliptical region:

```
HDC hdc;
HRGN hrgn;

hrgn = CreateEllipticRgn(10, 10, 110, 110);
SelectObject(hdc, hrgn);
PaintRgn(hdc, hrgn);

DeleteObject(hrgn);
```

### See Also

CreateBrushIndirect, CreateDIBPatternBrush, CreateHatchBrush, CreatePatternBrush, CreateSolidBrush, FillRgn

## PatBlt (2.x)

```
BOOL PatBlt(hdc, nLeftRect, nTopRect, nwidth, nheight, fdwRop)
HDC hdc; /* handle of device context */
int nLeftRect; /* x-
coordinate top-left corner destination rectangle
/
int nTopRect;
* y-coordinate top-left corner destination rectangle
/
int nwidth;
* width of destination rectangle
/
int nheight;
* height of destination rectangle
/
DWORD fdwRop;
* raster operation
/
```

The PatBlt function creates a bit pattern on the specified device. The pattern is a combination of the selected brush and the pattern already on the device. The specified raster-operation code defines how the patterns are combined.

Parameter	Description														
hdc	Identifies the device context.														
nLeftRect	Specifies the logical x-coordinate of the upper-left corner of the rectangle that receives the pattern.														
nTopRect	Specifies the logical y-coordinate of the upper-left corner of the rectangle that receives the pattern.														
nwidth	Specifies the width, in logical units, of the rectangle that will receive the pattern.														
nheight	Specifies the height, in logical units, of the rectangle that will receive the pattern.														
fdwRop	Specifies the raster-operation code that determines how the graphics device interface (GDI) combines the colors in the output operation. This parameter can be one of the following values:														
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>PATCOPY</td><td>Copies the pattern to the destination bitmap.</td></tr><tr><td>PATINVERT</td><td>Combines the destination bitmap with the pattern by using the Boolean XOR operator.</td></tr><tr><td>PATPAINT</td><td>Paints the destination bitmap.</td></tr><tr><td>DSTINVERT</td><td>Inverts the destination bitmap.</td></tr><tr><td>BLACKNESS</td><td>Turns all output black.</td></tr><tr><td>WHITENESS</td><td>Turns all output white.</td></tr></tbody></table>	Value	Meaning	PATCOPY	Copies the pattern to the destination bitmap.	PATINVERT	Combines the destination bitmap with the pattern by using the Boolean XOR operator.	PATPAINT	Paints the destination bitmap.	DSTINVERT	Inverts the destination bitmap.	BLACKNESS	Turns all output black.	WHITENESS	Turns all output white.
Value	Meaning														
PATCOPY	Copies the pattern to the destination bitmap.														
PATINVERT	Combines the destination bitmap with the pattern by using the Boolean XOR operator.														
PATPAINT	Paints the destination bitmap.														
DSTINVERT	Inverts the destination bitmap.														
BLACKNESS	Turns all output black.														
WHITENESS	Turns all output white.														

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The raster operations listed for this function are a limited subset of the full 256 ternary raster-operation codes; in particular, a raster-operation code that refers to a source cannot be used.

Not all devices support the PatBlt function. To determine whether a device supports PatBlt, an application can call the [GetDeviceCaps](#) function with the RASTERCAPS index.

### Example

The following example uses the [CreateBitmap](#) function to create a bitmap with a zig-zag pattern, and then uses the PatBlt function to fill the client area with that pattern:

```
HDC hdc;
HBITMAP hbmp;
```

```
HBRUSH hbr, hbrPrevious;
RECT rc;

int aZigzag[] = { 0xFF, 0xF7, 0xEB, 0xDD, 0xBE, 0x7F, 0xFF, 0xFF };

hbmp = CreateBitmap(8, 8, 1, 1, aZigzag);
hbr = CreatePatternBrush(hbmp);

hdc = GetDC(hwnd);
UnrealizeObject(hbr);
hbrPrevious = SelectObject(hdc, hbr);
GetClientRect(hwnd, &rc);

PatBlt(hdc, rc.left, rc.top,
        rc.right - rc.left, rc.bottom - rc.top, PATCOPY);
SelectObject(hdc, hbrPrevious);
ReleaseDC(hwnd, hdc);

DeleteObject(hbr);
DeleteObject(hbmp);

See Also
GetDeviceCaps
```

Windows 3.1 corrections

The following raster operation has been added:

Value	Meaning
PATPAINT	Paints the destination bitmap.



## Pie (2.x)

```
BOOL Pie(hdc, nLeftRect, nTopRect, nRightRect, nBottomRect, nxStartArc, nyStartArc, nxEndArc,
        nyEndArc)
HDC hdc;          /* handle of device context */
int nLeftRect;
/* x-coordinate upper-left corner bounding rectangle
/
int nTopRect;
/* y-coordinate upper-left corner bounding rectangle
/
int nRightRect;
/* x-coordinate lower-right corner bounding rectangle
/
int nBottomRect;
/* y-coordinate lower-right corner bounding rectangle
/
int nxStartArc;
/* x-coordinate arc starting point
/
int nyStartArc;
/* y-coordinate arc starting point
/
int nxEndArc;
/* x-coordinate arc ending point
/
int nyEndArc;
/* y-coordinate arc ending point
/
```

The Pie function draws a pie-shaped wedge by drawing an elliptical arc whose center and two endpoints are joined by lines.

Parameter	Description
hdc	Identifies the device context.
nLeftRect	Specifies the logical x-coordinate of the upper-left corner of the bounding rectangle.
nTopRect	Specifies the logical y-coordinate of the upper-left corner of the bounding rectangle.
nRightRect	Specifies the logical x-coordinate of the lower-right corner of the bounding rectangle.
nBottomRect	Specifies the logical y-coordinate of the lower-right corner of the bounding rectangle.
nxStartArc	Specifies the logical x-coordinate of the arc's starting point. This point does not have to lie exactly on the arc.
nyStartArc	Specifies the logical y-coordinate of the arc's starting point. This point does not have to lie exactly on the arc.
nxEndArc	Specifies the logical x-coordinate of the arc's endpoint. This point does not have to lie exactly on the arc.
nyEndArc	Specifies the logical y-coordinate of the arc's endpoint. This point does not have to lie exactly on the arc.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The center of the arc drawn by the Pie function is the center of the bounding rectangle specified by the nLeftRect, nTopRect, nRightRect, and nBottomRect parameters. The starting and ending points of the arc are specified by the nxStartArc, nyStartArc, nxEndArc, and nyEndArc parameters. The function draws the arc by using the selected pen, moving in a counterclockwise direction. It then draws two additional lines from each endpoint to the arc's center. Finally, it fills the pie-shaped area by using the current brush.

If nxStartArc equals nxEndArc and nyStartArc equals nyEndArc, the result is an ellipse with a single line from the center of the ellipse to the point (nxStartArc,nyStartArc) or (nxEndArc,nyEndArc).

The figure drawn by this function extends up to but does not include the right and bottom coordinates.

This means that the height of the figure is  $nBottomRect - nTopRect$  and the width of the figure is  $nRightRect - nLeftRect$ .

Both the width and the height of a rectangle must be greater than 2 units and less than 32,767 units.

#### Example

The following example uses a RECT structure to store the points that define the bounding rectangle and uses POINT structures to store the coordinates that specify the beginning and end of the wedge:

```
HDC hdc;
```

```
RECT rc = { 10, 10, 180, 140 };
```

```
POINT ptStart = { 12, 12 };
```

```
POINT ptEnd = { 128, 135 };
```

```
Pie(hdc, rc.left, rc.top, rc.right, rc.bottom,  
    ptStart.x, ptStart.y, ptEnd.x, ptEnd.y);
```

See Also

Chord, POINT, RECT

## PlayMetaFile (2.x)

```
BOOL PlayMetaFile(hdc, hmf)
HDC hdc;          /* handle of device context */
HMETAFILE hmf;    /* handle of metafile */
/
```

The PlayMetaFile function plays the contents of the specified metafile on the given device. The metafile can be played any number of times.

Parameter	Description
hdc	Identifies the device context of the output device.
hmf	Identifies the metafile to be played.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Example

The following example uses the CreateMetaFile function to create a device-context handle of a memory metafile, draws a line in the device context, retrieves a metafile handle by calling the CloseMetaFile function, plays the metafile by using the PlayMetaFile function, and finally deletes the metafile by using the DeleteMetaFile function:

```
HDC hdcMeta;
HMETAFILE hmf;

hdcMeta = CreateMetaFile(NULL);
MoveTo(hdcMeta, 10, 10);
LineTo(hdcMeta, 100, 100);
hmf = CloseMetaFile(hdcMeta);
PlayMetaFile(hdc, hmf);
DeleteMetaFile(hmf);
```

### See Also

PlayMetaFileRecord

## PlayMetaFileRecord (2.x)

```
void PlayMetaFileRecord(hdc, lpht, lpmr, cHandles)
HDC hdc; /* handle of device context */
HANDLETABLE FAR* lpht;
/* address of table of object handles
/
METARECORD FAR* lpmr;
/* address of metafile record
/
UINT cHandles;
/* number of handles in table
/
```

The PlayMetaFileRecord function plays a metafile record by executing the graphics device interface (GDI) function contained in the record.

Parameter	Description
hdc	Identifies the device context of the output device.
lpht	Points to a table of handles associated with the objects (pens, brushes, and so on) in the metafile.
lpmr	Points to the metafile record to be played.
cHandles	Specifies the number of handles in the handle table.

### Returns

This function does not return a value.

### Comments

An application typically uses this function in conjunction with the EnumMetafile function to modify and then play a metafile.

### Example

The following example creates a dashed green pen and passes it to the callback function for the EnumMetaFile function. If the first element in the array of object handles contains a handle, that handle is replaced by the handle of the green pen before the PlayMetaFileRecord function is called. (For this example, it is assumed that the table of object handles contains only one handle and that it is a pen handle.)

```
MFENUMPROC lpEnumMetaProc;
HPEN hpenGreen;

lpEnumMetaProc = (MFENUMPROC) MakeProcInstance(
    (FARPROC) EnumMetaFileProc, hAppInstance);
hpenGreen = CreatePen(PS_DASH, 1, RGB(0, 255, 0));
EnumMetaFile(hdc, hmf, lpEnumMetaProc, (LPARAM) &hpenGreen);
FreeProcInstance(FARPROC) lpEnumMetaProc);
DeleteObject(hpenGreen);
.
.
.

int FAR PASCAL EnumMetaFileProc(HDC hdc, HANDLETABLE FAR* lpHTable,
METARECORD FAR* lpMFR, int cObj, BYTE FAR* lpClientData)
{
    if (lpHTable->objectHandle[0] != 0)
        lpHTable->objectHandle[0] = *(HPEN FAR *) lpClientData;
        PlayMetaFileRecord(hdc, lpHTable, lpMFR, cObj);

    return 1;
}
```

### See Also

EnumMetafile, PlayMetaFile

## Polygon (2.x)

```
BOOL Polygon(hdc, lppt, cPoints)
HDC hdc; /* handle of device context */
const POINT FAR* lppt;
/* address of array with points for vertices */
/
int cPoints;
/* number of points in array */
/
```

The Polygon function draws a polygon consisting of two or more points (vertices) connected by lines. The system closes the polygon automatically, if necessary, by drawing a line from the last vertex to the first. Polygons are surrounded by a frame drawn by using the current pen and filled by using the current brush.

Parameter	Description
hdc	Identifies the device context.
lppt	Points to an array of <u>POINT</u> structures that specify the vertices of the polygon. Each structure in the array specifies a vertex.
cPoints	Specifies the number of vertices in the array.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The current polygon-filling mode can be retrieved or set by using the GetPolyFillMode and SetPolyFillMode functions.

### Example

The following example assigns values to an array of points and then calls the Polygon function:

```
HDC hdc;
```

```
POINT aPoints[3];
```

```
aPoints[0].x = 50;
aPoints[0].y = 10;
aPoints[1].x = 250;
aPoints[1].y = 50;
aPoints[2].x = 125;
aPoints[2].y = 130;
```

```
Polygon(hdc, aPoints, sizeof(aPoints) / sizeof(POINT));
```

### See Also

GetPolyFillMode, Polyline, PolyPolygon, SetPolyFillMode, POINT

## Polyline (2.x)

```
BOOL Polyline(hdc, lppt, cPoints)
HDC hdc; /* handle of device context */
const POINT FAR* lppt;
/* address of array with points to connect */
/
int cPoints;
/* number of points in array */
/
```

The Polyline function draws a set of line segments, connecting the specified points. The lines are drawn from the first point through subsequent points, using the current pen. Unlike the LineTo function, the Polyline function neither uses nor updates the current position.

Parameter	Description
hdc	Identifies the device context.
lppt	Points to an array of <u>POINT</u> structures. Each structure in the array specifies a point.
cPoints	Specifies the number of points in the array. This value must be at least 2.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Example

The following example assigns values to an array of points and then calls the Polyline function:

```
HDC hdc;
```

```
POINT aPoints[3];
```

```
aPoints[0].x = 50;
aPoints[0].y = 10;
aPoints[1].x = 250;
aPoints[1].y = 50;
aPoints[2].x = 125;
aPoints[2].y = 130;
```

```
Polyline(hdc, aPoints, sizeof(aPoints) / sizeof(POINT));
```

See Also

LineTo, Polygon, POINT

## PolyPolygon (3.0)

```
BOOL PolyPolygon(hdc, lppt, lpnPolyCounts, cPolygons)
HDC hdc; /* handle of device context */
const POINT FAR* lppt;
/* address of array with vertices
/
int FAR* lpnPolyCounts;
/* address of array with point counts
/
int cPolygons;
/* number of polygons to draw
/
```

The PolyPolygon function creates two or more polygons that are filled by using the current polygon-filling mode. The polygons may be disjoint or overlapping.

Parameter	Description
hdc	Identifies the device context.
lppt	Points to an array of <u>POINT</u> structures. Each structure in the array specifies a vertex of a polygon.
lpnPolyCounts	Points to an array of integers, each of which specifies the number of points in one of the polygons in the array pointed to by the lppt parameter.
cPolygons	Specifies the number of polygons to be drawn. This value must be at least 2.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

Each polygon specified in a call to the PolyPolygon function must be closed. Unlike polygons created by the Polygon function, the polygons created by PolyPolygon are not closed automatically.

The PolyPolygon function creates two or more polygons. To create a single polygon, an application should use the Polygon function.

The current polygon-filling mode can be retrieved or set by using the GetPolyFillMode and SetPolyFillMode functions.

### Example

The following example draws two overlapping polygons by assigning values to an array of points and then calling the PolyPolygon function:

```
HDC hdc;

POINT aPolyPoints[8];
int aVertices[] = { 4, 4 };

aPolyPoints[0].x = 50;
aPolyPoints[0].y = 10;
aPolyPoints[1].x = 250;
aPolyPoints[1].y = 50;
aPolyPoints[2].x = 125;
aPolyPoints[2].y = 130;
aPolyPoints[3].x = 50;
aPolyPoints[3].y = 10;

aPolyPoints[4].x = 100;
aPolyPoints[4].y = 25;
aPolyPoints[5].x = 300;
aPolyPoints[5].y = 125;
aPolyPoints[6].x = 70;
aPolyPoints[6].y = 150;
aPolyPoints[7].x = 100;
aPolyPoints[7].y = 25;
```

```
PolyPolygon(hdc, aPolyPoints, aVertices,  
            sizeof(aVertices) / sizeof(int));
```

See Also

[GetPolyFillMode](#), [Polygon](#), [Polyline](#), [SetPolyFillMode](#), [POINT](#)



## PtInRegion (2.x)

```
BOOL PtInRegion(hrgn, nXPos, nYPos)
HRGN hrgn;      /* handle of region      */
int nXPos;      /* x-coordinate of point */
int nYPos;      /* y-coordinate of point */
/
```

The PtInRegion function determines whether a specified point is in the given region.

Parameter	Description
hrgn	Identifies the region to be examined.
nXPos	Specifies the logical x-coordinate of the point.
nYPos	Specifies the logical y-coordinate of the point.

### Returns

The return value is nonzero if the point is in the region. Otherwise, it is zero.

### Example

The following example uses the PtInRegion function to determine whether the point (50, 50) is in the specified region and prints the result:

```
HRGN hrgn;
BOOL fPtIn;
LPSTR lpszInRegion = "Specified point is in region.";
LPSTR lpszNotInRegion = "Specified point is not in region.";

fPtIn = PtInRegion(hrgn, 50, 50);
if (!fPtIn)
    TextOut(hdc, 10, 10, lpszNotInRegion,
            lstrlen(lpszNotInRegion));
else
    TextOut(hdc, 10, 10, lpszInRegion, lstrlen(lpszInRegion));
```

### See Also

[RectInRegion](#)

## PtVisible (2.x)

```
BOOL PtVisible(hdc, nXPos, nYPos)
HDC hdc;          /* handle of device context */
int nXPos;        /* x-coordinate of point to query */
/
int nYPos;
/* y-coordinate of point to query
/
```

The PtVisible function determines whether the specified point is within the clipping region of the given device context.

Parameter	Description
hdc	Identifies the device context.
nXPos	Specifies the logical x-coordinate of the point.
nYPos	Specifies the logical y-coordinate of the point.

### Returns

The return is nonzero if the point is within the clipping region. Otherwise, it is zero.

### Example

The following example creates a rectangular region, displays a message inside it, and selects the region as the clipping region. The PtVisible function is used to determine whether coordinates generated by a double-click are inside the region. If so, the message changes to "Thank you." If not, the CombineRgn function is used to create a clipping region that combines the first region with a new region that surrounds the specified coordinates, and the word "Missed!" is displayed at the coordinates.

```
HDC hdcLocal;
HRGN hrgnClick, hrgnMiss, hrgnCombine;
HBRUSH hbr;

hdcLocal = GetDC(hwnd);
hbr = GetStockObject(BLACK_BRUSH);

hrgnClick = CreateRectRgn(90, 95, 225, 120);
FrameRgn(hdcLocal, hrgnClick, hbr, 1, 1);
TextOut(hdcLocal, 100, 100, "Double-click here.", 18);
SelectClipRgn(hdcLocal, hrgnClick);

if (PtVisible(hdcLocal, XClick, YClick)) {
    PaintRgn(hdcLocal, hrgnClick);
    FrameRgn(hdcLocal, hrgnClick, hbr, 1, 1);
    TextOut(hdcLocal, 100, 100, "Thank you.", 10);
}
else if (XClick > 0) {
    hrgnMiss = CreateRectRgn(XClick - 5, YClick - 5, XClick + 60,
    YClick + 20);
    hrgnCombine = CreateRectRgn(0, 0, 0, 0);
    CombineRgn(hrgnCombine, hrgnClick, hrgnMiss, RGN_OR);
    SelectClipRgn(hdcLocal, hrgnCombine);
    FrameRgn(hdcLocal, hrgnCombine, hbr, 1, 1);
    TextOut(hdcLocal, XClick, YClick, "Missed!", 7);
}

InvalidateRect(hwnd, NULL, FALSE);

DeleteObject(hrgnClick);
DeleteObject(hrgnMiss);
DeleteObject(hrgnCombine);
ReleaseDC(hwnd, hdcLocal);
```

### See Also

CombineRgn, RectVisible

### QueryAbort (3.1)

```
BOOL QueryAbort(hdc, reserved)
HDC hdc;          /* device-context handle */
int reserved;    /* reserved; must be zero */
```

The QueryAbort function calls the AbortProc callback function for a printing application and queries whether the printing should be terminated.

Parameter	Description
hdc	Identifies the device context.
reserved	Reserved; must be zero.

#### Returns

The return value is TRUE if printing should continue or if there is no abort procedure. It is FALSE if the print job should be terminated. The return value is supplied by the AbortProc callback function.

#### See Also

AbortDoc, AbortProc, SetAbortProc

## Rectangle (2.x)

```
BOOL Rectangle(hdc, nLeftRect, nTopRect, nRightRect, nBottomRect)
HDC hdc;          /* handle of device context */
int nLeftRect;    /* x-coordinate upper-
left corner
/
int nTopRect;     /* y-coordinate upper-left corner
/
int nRightRect;   /* x-coordinate lower-right corner
/
int nBottomRect; /* y-coordinate lower-right corner
/
```

The Rectangle function draws a rectangle, using the current pen. The interior of the rectangle is filled by using the current brush.

Parameter	Description
hdc	Identifies the device context.
nLeftRect	Specifies the logical x-coordinate of the upper-left corner of the rectangle.
nTopRect	Specifies the logical y-coordinate of the upper-left corner of the rectangle.
nRightRect	Specifies the logical x-coordinate of the lower-right corner of the rectangle.
nBottomRect	Specifies the logical y-coordinate of the lower-right corner of the rectangle.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The figure this function draws extends up to, but does not include, the right and bottom coordinates. This means that the height of the figure is nBottomRect - nTopRect and the width of the figure is nRightRect - nLeftRect.

Both the width and the height of a rectangle must be greater than 2 units and less than 32,767 units.

### Example

The following example uses a RECT structure to store the coordinates used by the Rectangle function:

```
HDC hdc;
RECT rc = { 10, 10, 180, 140 };
Rectangle(hdc, rc.left, rc.top,
rc.right, rc.bottom);
```

### See Also

PolyLine, RoundRect, RECT

### RectInRegion (3.0)

```
BOOL RectInRegion(hrgn, lprc)
HRGN hrgn; /* handle of region */
const RECT FAR* lprc; /* address of structure with rectangle */
/
```

The RectInRegion function determines whether any part of the specified rectangle is within the boundaries of the given region.

Parameter	Description
hrgn	Identifies the region.
lprc	Points to a <u>RECT</u> structure containing the coordinates of the rectangle.

#### Returns

The return value is nonzero if any part of the specified rectangle lies within the boundaries of the region. Otherwise, it is zero.

#### Example

The following example uses the RectInRegion function to determine whether a specified rectangle is in a region and prints the result:

```
HRGN hrgn;
RECT rc = { 100, 10, 130, 50 };
BOOL fRectIn;
LPSTR lpszOverlap = "Some overlap between rc and region.";
LPSTR lpszNoOverlap = "No common points in rc and region.";

fRectIn = RectInRegion(hrgn, &rc);
if (!fRectIn)
    TextOut(hdc, 10, 10, lpszNoOverlap, lstrlen(lpszNoOverlap));
else
    TextOut(hdc, 10, 10, lpszOverlap, lstrlen(lpszOverlap));
```

#### See Also

PtInRegion, RECT

## RectVisible (2.x)

```
BOOL RectVisible(hdc, lprc)
HDC hdc; /* handle of device context */
const RECT FAR* lprc; /* address of structure with rectangle */
/
```

The RectVisible function determines whether any part of the specified rectangle lies within the clipping region of the given device context.

Parameter	Description
hdc	Identifies the device context.
lprc	Points to a <u>RECT</u> structure that contains the logical coordinates of the specified rectangle.

### Returns

The return value is nonzero if some portion of the rectangle is within the clipping region. Otherwise, it is zero.

### Example

The following example paints a clipping region yellow by painting the client area. The RectVisible function is called to determine whether a specified rectangle overlaps the clipping region. If there is some overlap, the rectangle is filled by using a red brush. If there is no overlap, text is displayed inside the clipping region. In this case, the rectangle and the region do not overlap, even though they both specify 110 as a boundary on the y-axis, because regions are defined as including the pixels up to but not including the specified right and bottom coordinates.

```
RECT rc, rcVis;
HRGN hrgn;
HBRUSH hbrRed, hbrYellow;

GetClientRect(hwnd, &rc);
hrgn = CreateRectRgn(10, 10, 310, 110);
SelectClipRgn(hdc, hrgn);

hbrYellow = CreateSolidBrush(RGB(255, 255, 0));
FillRect(hdc, &rc, hbrYellow);

SetRect(&rcVis, 10, 110, 310, 300);
if (RectVisible(hdc, &rcVis)) {
    hbrRed = CreateSolidBrush(RGB(255, 0, 0));
    FillRect(hdc, &rcVis, hbrRed);
    DeleteObject(hbrRed);
}
else {
    SetBkColor(hdc, RGB(255, 255, 0));
    TextOut(hdc, 20, 50, "Rectangle outside clipping region.", 34);
}

DeleteObject(hbrYellow);
DeleteObject(hrgn);
```

### See Also

CreateRectRgn, PtVisible, SelectClipRgn, RECT

## RemoveFontResource (2.x)

BOOL RemoveFontResource(lpszFile)  
LPCSTR lpszFile; /\* address of string for filename \*/

The RemoveFontResource function removes an added font resource from the specified file or from the Windows font table.

Parameter	Description
lpszFile	Points to a string that names the font resource file or contains a handle of a loaded module. If this parameter points to the font resource file, the string must be null-terminated and have the MS-DOS filename format. If the parameter contains a handle, the handle must be in the low-order word and the high-order word must be zero.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

Any application that adds or removes fonts from the Windows font table should send a WM\_FONTCHANGE message to all top-level windows in the system by using the SendMessage function with the hwnd parameter set to 0xFFFF.

In some cases, the RemoveFontResource function may not remove the font resource immediately. If there are outstanding references to the resource, it remains loaded until the last logical font using it has been removed (deleted) by using the DeleteObject function.

### Example

The following example uses the AddFontResource function to add a font resource from a file, notifies other applications by using the SendMessage function, then removes the font resource by calling the RemoveFontResource function:

```
AddFontResource("fontres.fon");  
SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);  
.  
. /* Work with the font. */  
.  
if (RemoveFontResource("fontres.fon")) {  
    SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);  
    return TRUE;  
}  
else  
    return FALSE;
```

### See Also

AddFontResource, DeleteObject, SendMessage

## ResetDC (3.1)

```
#include <print.h>
```

```
HDC ResetDC(hdc, lpdm)
```

```
HDC hdc; /* handle of device context */
```

```
const DEVMODE FAR* lpdm;
```

```
* address of DEVMODE structure
```

```
/
```

The ResetDC function updates the given device context, based on the information in the specified DEVMODE structure.

Parameter	Description
hdc	Identifies the device context to be updated.
lpdm	Points to a <u>DEVMODE</u> structure containing information about the new device context.

### Returns

The return value is the handle of the original device context if the function is successful. Otherwise, it is NULL.

### Comments

An application will typically use the ResetDC function when a window receives a WM\_DEVMODECHANGE message. ResetDC can also be used to change the paper orientation or paper bins while printing a document.

The ResetDC function cannot be used to change the driver name, device name or the output port. When the user changes the port connection or device name, the application must delete the original device context and create a new device context with the new information.

Before calling ResetDC, the application must ensure that all objects (other than stock objects) that had been selected into the device context have been selected out.

### See Also

DeviceCapabilities, Escape, ExtDeviceMode, DEVMODE, WM\_DEVMODECHANGE



### ResizePalette (3.0)

```
BOOL ResizePalette(hpal, cEntries)
HPALETTE hpal;          /* handle of palette */
UINT cEntries;          /*
number of palette entries after resizing
/
```

The ResizePalette function changes the size of the given logical palette.

Parameter	Description
hpal	Identifies the palette to be changed.
cEntries	Specifies the number of entries in the palette after it has been resized.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

If an application calls the ResizePalette function to reduce the size of the palette, the entries remaining in the resized palette are unchanged. If the application calls ResizePalette to enlarge the palette, the additional palette entries are set to black (the red, green, and blue values are all zero) and the flags for all additional entries are set to zero.

## RestoreDC (2.x)

```
BOOL RestoreDC(hdc, nSavedDC)
HDC hdc;          /* handle of device context          */
int nSavedDC;     /* integer
identifying device context to restore
/
```

The RestoreDC function restores the given device context to a previous state. The device context is restored by popping state information off a stack created by earlier calls to the SaveDC function.

Parameter	Description
hdc	Identifies the device context.
nSavedDC	Specifies the device context to be restored. This parameter can be a value returned by a previous <u>SaveDC</u> function. If the parameter is -1, the most recently saved device context is restored.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The stack can contain the state information for several instances of the device context. If the context specified by the nSavedDC parameter is not at the top of the stack, RestoreDC deletes all state information between the instance specified by nSavedDC and the top of the stack.

### Example

The following example uses the GetMapMode function to retrieve the mapping mode for the current device context, uses the SaveDC function to save the state of the device context, changes the mapping mode, restores the previous state of the device context by using the RestoreDC function, and retrieves the mapping mode again. The final mapping mode is the same as the mapping mode prior to the call to the SaveDC function.

```
HDC hdcLocal;
int MapMode;
char *aModes[] = {"ZERO", "MM_TEXT", "MM_LOMETRIC", "MM_HIMETRIC",
                  "MM_LOENGLISH", "MM_HIENGLISH", "MM_TWIPS",
                  "MM_ISOTROPIC", "MM_ANISOTROPIC" };
```

```
hdcLocal = GetDC(hwnd);
MapMode = GetMapMode(hdcLocal);
TextOut(hdc, 100, 100, (LPSTR) aModes[MapMode],
         lstrlen(aModes[MapMode]));
```

```
SaveDC(hdcLocal);
```

```
SetMapMode(hdcLocal, MM_LOENGLISH);
MapMode = GetMapMode(hdcLocal);
TextOut(hdc, 100, 120, (LPSTR) aModes[MapMode],
         lstrlen(aModes[MapMode]));
```

```
RestoreDC(hdcLocal, -1);
```

```
MapMode = GetMapMode(hdcLocal);
TextOut(hdc, 100, 140, (LPSTR) aModes[MapMode],
         lstrlen(aModes[MapMode]));
```

```
ReleaseDC(hwnd, hdcLocal);
```

See Also

SaveDC

## RoundRect (2.x)

```
BOOL RoundRect(hdc, nLeftRect, nTopRect, nRightRect, nBottomRect, nEllipseWidth, nEllipseHeight)
HDC hdc; /* handle of device context */
int nLeftRect; /* x-coordinate
upper-left corner
/
int nTopRect;
* y-coordinate upper-left corner
/
int nRightRect;
* x-coordinate lower-right corner
/
int nBottomRect;
* y-coordinate lower-right corner
/
int nEllipseWidth;
* width of ellipse for rounded corners
/
int nEllipseHeight;
* height of ellipse for rounded corners
/
```

The RoundRect function draws a rectangle with rounded corners, using the current pen. The interior of the rectangle is filled by using the current brush.

Parameter	Description
hdc	Identifies the device context.
nLeftRect	Specifies the logical x-coordinate of the upper-left corner of the rectangle.
nTopRect	Specifies the logical y-coordinate of the upper-left corner of the rectangle.
nRightRect	Specifies the logical x-coordinate of the lower-right corner of the rectangle.
nBottomRect	Specifies the logical y-coordinate of the lower-right corner of the rectangle.
nEllipseWidth	Specifies the width, in logical units, of the ellipse used to draw the rounded corners.
nEllipseHeight	Specifies the height, in logical units, of the ellipse used to draw the rounded corners.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The figure this function draws extends up to but does not include the right and bottom coordinates. This means that the height of the figure is nBottomRect - nTopRect and the width of the figure is nRightRect - nLeftRect.

Both the width and the height of a rectangle must be greater than 2 units and less than 32,767 units.

### Example

The following example uses a RECT structure to store the coordinates used by the RoundRect function:

```
HDC hdc;

RECT rc = { 10, 10, 180, 140 };
int iEllipseWidth, iEllipseHeight;

iEllipseWidth = 20;
iEllipseHeight = 40;

RoundRect(hdc, rc.left, rc.top, rc.right, rc.bottom,
          iEllipseWidth, iEllipseHeight);
```

### See Also

Rectangle, RECT

## SaveDC (2.x)

```
int SaveDC(hdc)
HDC hdc;      /* handle of device context */
```

The SaveDC function saves the current state of the given device context by copying state information (such as clipping region, selected objects, and mapping mode) to a context stack. The saved device context can later be restored by using the RestoreDC function.

Parameter	Description
hdc	Identifies the device context to be saved.

### Returns

The return value is an integer identifying the saved device context if the function is successful. This integer can be used to restore the device context by calling the RestoreDC function. The return value is zero if an error occurs.

### Comments

The SaveDC function can be used any number of times to save any number of device-context states.

### Example

The following example uses the GetMapMode function to retrieve the mapping mode for the current device context, uses the SaveDC function to save the state of the device context, changes the mapping mode, restores the previous state of the device context by using the RestoreDC function, and retrieves the mapping mode again. The final mapping mode is the same as the mapping mode prior to the call to the SaveDC function.

```
HDC hdcLocal;
int MapMode;
char *aModes[] = {"ZERO", "MM_TEXT", "MM_LOMETRIC", "MM_HIMETRIC",
                  "MM_LOENGLISH", "MM_HIENGLISH", "MM_TWIPS",
                  "MM_ISOTROPIC", "MM_ANISOTROPIC" };

hdcLocal = GetDC(hwnd);
MapMode = GetMapMode(hdcLocal);
TextOut(hdc, 100, 100, (LPSTR) aModes[MapMode],
         lstrlen(aModes[MapMode]));

SaveDC(hdcLocal);

SetMapMode(hdcLocal, MM_LOENGLISH);
MapMode = GetMapMode(hdcLocal);
TextOut(hdc, 100, 120, (LPSTR) aModes[MapMode],
         lstrlen(aModes[MapMode]));

RestoreDC(hdcLocal, -1);

MapMode = GetMapMode(hdcLocal);
TextOut(hdc, 100, 140, (LPSTR) aModes[MapMode],
         lstrlen(aModes[MapMode]));

ReleaseDC(hwnd, hdcLocal);
```

See Also  
RestoreDC

## ScaleViewportExt (2.x)

```
DWORD ScaleViewportExt(hdc, nXNum, nXDenom, nYNum, nYDenom)
HDC hdc; /* handle of device context */
int nXNum; /* amount by
which current x-extent is multiplied
/
int nXDenom;
* amount by which current x-extent is divided
/
int nYNum;
* amount by which current y-extent is multiplied
/
int nYDenom;
* amount by which current y-extent is divided
/
```

The ScaleViewportExt function modifies the viewport extents relative to the current values.

Parameter	Description
hdc	Identifies the device context.
nXNum	Specifies the amount by which to multiply the current x-extent.
nXDenom	Specifies the amount by which to divide the result of multiplying the current x-extent by the value of the nXNum parameter.
nYNum	Specifies the amount by which to multiply the current y-extent.
nYDenom	Specifies the amount by which to divide the result of multiplying the current y-extent by the value of the nYNum parameter.

### Returns

The low-order word of the return value contains the x-extent, in device units, of the previous viewport if the function is successful; the high-order word contains the y-extent.

### Comments

The new viewport extents are calculated by multiplying the current extents by the given numerator and then dividing by the given denominator, as shown in the following formulas:

$$\text{nXNewVE} = (\text{nXOldVE} * \text{nXNum}) / \text{nXDenom}$$
$$\text{nYNewVE} = (\text{nYOldVE} * \text{nYNum}) / \text{nYDenom}$$

### Example

The following example draws a rectangle that is 4 logical units high and 4 logical units wide. It then calls the ScaleViewportExt function and draws a rectangle that is 8 units by 8 units. Because of the viewport scaling, the second rectangle is the same size as the first.

```
HDC hdc;
RECT rc;

GetClientRect(hwnd, &rc);
hdc = GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);

SetWindowExt(hdc, 10, 10);
SetViewportExt(hdc, rc.right, rc.bottom);
Rectangle(hdc, 3, 3, 7, 7);

ScaleViewportExt(hdc, 1, 2, 1, 2);
Rectangle(hdc, 6, 6, 14, 14);

ReleaseDC(hwnd, hdc);
```

See Also  
[GetViewportExt](#)

### ScaleViewportExtEx (3.1)

```
BOOL ScaleViewportExtEx(hdc, nXnum, nXdenom, nYnum, nYdenom, lpSize)
HDC hdc; /* handle of device context */
int nXnum; /* amount by which current x-extent is multiplied */
/* amount by which current x-extent is multiplied */
/
int nXdenom; /* amount by which current x-extent is divided */
/* amount by which current x-extent is divided */
/
int nYnum; /* amount by which current y-extent is multiplied */
/* amount by which current y-extent is multiplied */
/
int nYdenom; /* amount by which current y-extent is divided */
/* amount by which current y-extent is divided */
/
SIZE FAR* lpSize; /* address of SIZE structure */
/* address of SIZE structure */
/
```

The ScaleViewportExtEx function modifies the viewport extents relative to the current values. The formulas are written as follows:

$$x_{\text{NewVE}} = (x_{\text{OldVE}} * X_{\text{num}}) / X_{\text{denom}}$$
$$y_{\text{NewVE}} = (y_{\text{OldVE}} * Y_{\text{num}}) / Y_{\text{denom}}$$

The new extent is calculated by multiplying the current extents by the given numerator and then dividing by the given denominator.

Parameter	Description
hdc	Identifies the device context.
nXnum	Specifies the amount by which to multiply the current x-extent.
nXdenom	Specifies the amount by which to divide the current x-extent.
nYnum	Specifies the amount by which to multiply the current y-extent.
nYdenom	Specifies the amount by which to divide the current y-extent.
lpSize	Points to a <u>SIZE</u> structure. The previous viewport extents, in device units, are placed in this structure. If lpSize is NULL, nothing is returned.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

## ScaleWindowExt (2.x)

```
DWORD ScaleWindowExt(hdc, nXNum, nXDenom, nYNum, nYDenom)
HDC hdc;          /* handle of device context */
int nXNum;        /* amount by
which current x-extent is multiplied
/
int nXDenom;
* amount by which current x-extent is divided
/
int nYNum;
* amount by which current y-extent is multiplied
/
int nYDenom;
* amount by which current y-extent is divided
/
```

The ScaleWindowExt function modifies the window extents relative to the current values.

Parameter	Description
hdc	Identifies the device context.
nXNum	Specifies the amount by which to multiply the current x-extent.
nXDenom	Specifies the amount by which to divide the result of multiplying the current x-extent by the value of the nXNum parameter.
nYNum	Specifies the amount by which to multiply the current y-extent.
nYDenom	Specifies the amount by which to divide the result of multiplying the current y-extent by the value of the nYNum parameter.

### Returns

The low-order word of the return value contains the x-extent, in logical units, of the previous window, if the function is successful; the high-order word contains the y-extent.

### Comments

The new window extents are calculated by multiplying the current extents by the given numerator and then dividing by the given denominator, as shown in the following formulas:

$$\text{nXNewWE} = (\text{nXOldWE} * \text{nXNum}) / \text{nXDenom}$$
$$\text{nYNewWE} = (\text{nYOldWE} * \text{nYNum}) / \text{nYDenom}$$

### Example

The following example draws a rectangle that is 4 logical units high and 4 logical units wide. It then calls the ScaleWindowExt function and draws a rectangle that is 8 units by 8 units. Because of the window scaling, the second rectangle is the same size as the first.

```
HDC hdc;
RECT rc;

GetClientRect(hwnd, &rc);
hdc = GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);

SetWindowExt(hdc, 10, 10);
SetViewportExt(hdc, rc.right, rc.bottom);
Rectangle(hdc, 3, 3, 7, 7);

ScaleWindowExt(hdc, 2, 1, 2, 1);
Rectangle(hdc, 6, 6, 14, 14);

ReleaseDC(hwnd, hdc);
```

See Also  
[GetWindowExt](#)

### ScaleWindowExtEx (3.1)

```
BOOL ScaleWindowExtEx(hdc, nXnum, nXdenom, nYnum, nYdenom, lpSize)
HDC hdc; /* handle of device context */
int nXnum; /* amount by which current x-extent is multiplied */
int nXdenom; /* amount by which current x-extent is divided */
int nYnum; /* amount by which current y-extent is multiplied */
int nYdenom; /* amount by which current y-extent is divided */
SIZE FAR* lpSize; /* address of SIZE structure */
```

The ScaleWindowExtEx function modifies the window extents relative to the current values. The formulas are written as follows:

$$x_{\text{NewWE}} = (x_{\text{OldWE}} * X_{\text{num}}) / X_{\text{denom}}$$
$$y_{\text{NewWE}} = (y_{\text{OldWE}} * Y_{\text{num}}) / Y_{\text{denom}}$$

The new extent is calculated by multiplying the current extents by the given numerator and then dividing by the given denominator.

Parameter	Description
hdc	Identifies the device context.
nXnum	Specifies the amount by which to multiply the current x-extent.
nXdenom	Specifies the amount by which to divide the current x-extent.
nYnum	Specifies the amount by which to multiply the current y-extent.
nYdenom	Specifies the amount by which to divide the current y-extent.
lpSize	Points to a <u>SIZE</u> structure. The previous window extents, in logical units, are placed in this structure. If lpSize is NULL, nothing is returned.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.



## SelectClipRgn (2.x)

```
int SelectClipRgn(hdc, hrgn)
HDC hdc;          /* handle of device context */
HRGN hrgn;        /* handle of region */
```

The SelectClipRgn function selects the given region as the current clipping region for the given device context.

Parameter	Description
hdc	Identifies the device context.
hrgn	Identifies the region to be selected. If this value is NULL, the entire client area is selected and output is still clipped to the window.

### Returns

The return value is SIMPLEREGION (region has no overlapping borders), COMPLEXREGION (region has overlapping borders), or NULLREGION (region is empty), if the function is successful. Otherwise, the return value is ERROR.

### Comments

The SelectClipRgn function selects only a copy of the specified region. Because SelectClipRgn uses only a copy, the region can be selected for any number of other device contexts or it can be deleted.

The coordinates for the specified region should be specified in device units.

Some printer devices support text output at a higher resolution than graphics output in order to retain the precision needed to express text metrics. These devices report device units at the higher resolution--that is, text units. These devices then scale coordinates for graphics so that several reported device units map to only one graphics unit. Applications should always call the SelectClipRgn function using the text unit. Applications that must take the scaling of graphics objects in the graphics device interface (GDI) can use the GETSCALINGFACTOR printer escape to determine the scaling factor. This scaling factor affects clipping. If a region is used to clip graphics, GDI divides the coordinates by the scaling factor. (If the region is used to clip text, however, GDI makes no scaling adjustment.) A scaling factor of 1 causes the coordinates to be divided by 2; a scaling factor of 2 causes the coordinates to be divided by 4; and so on.

### Example

The following example uses the GetClipBox function to determine the size of the current clipping region and the GetTextExtent function to determine the width of a line of text. If the text will not fit in the clipping region, the SelectClipRgn is used to make the region wide enough for the text. The output is clipped to the window regardless of the size of the region specified in the second parameter of SelectClipRegion.

```
HRGN hrgnClip;
RECT rcClip;
LPSTR lpszTest = "Test of clipping region.";
DWORD dwStringLen;
WORD wExtent;

GetClipBox(hdc, &rcClip);
dwStringLen = GetTextExtent(hdc, lpszTest, lstrlen(lpszTest));
wExtent = LOWORD(dwStringLen);
if (rcClip.right < 50 + wExtent) {
    hrgnClip = CreateRectRgn(50, 50, 50 + wExtent, 80);
    SelectClipRgn(hdc, hrgnClip);
}

TextOut(hdc, 50, 60, lpszTest, lstrlen(lpszTest));

DeleteObject(hrgnClip);
```

See Also

GetClipBox, GetTextExtent, GETSCALINGFACTOR

## SelectObject (2.x)

```
HGDIOBJ SelectObject(hdc, hgdiobj)
HDC hdc; /* handle of device context */
HGDIOBJ hgdiobj; /* handle of object */
/
```

The SelectObject function selects an object into the given device context. The new object replaces the previous object of the same type.

Parameter	Description
hdc	Identifies the device context.
hgdiobj	Identifies the object to be selected. The object can be one of the following and must have been created by using one of the listed functions:
Object	Functions
Bitmap	<u>CreateBitmap</u> , <u>CreateBitmapIndirect</u> , <u>CreateCompatibleBitmap</u> , <u>CreateDIBitmap</u>
Brush	<u>CreateBrushIndirect</u> , <u>CreateDIBPatternBrush</u> , <u>CreateHatchBrush</u> , <u>CreatePatternBrush</u> , <u>CreateSolidBrush</u>
Font	<u>CreateFont</u> , <u>CreateFontIndirect</u>
Pen	<u>CreatePen</u> , <u>CreatePenIndirect</u>
Region	<u>CreateEllipticRgn</u> , <u>CreateEllipticRgnIndirect</u> , <u>CreatePolygonRgn</u> , <u>CreateRoundRectRgn</u> , <u>CreateRectRgn</u> , <u>CreateRectRgnIndirect</u>

### Returns

The return value is the handle of the object being replaced, if the function is successful. Otherwise, it is NULL.

If the hgdiobj parameter identifies a region, this function performs the same task as the SelectClipRgn function and the return value is SIMPLEREGION (region has no overlapping borders), COMPLEXREGION (region has overlapping borders), or NULLREGION (region is empty). If an error occurs, the return value is ERROR and the previously selected object of the specified type remains selected in the device context.

### Comments

When an application uses the SelectObject function to select a font, pen, or brush, the system allocates space for that object in its data segment. Because data-segment space is limited, an application should use the DeleteObject function to remove each drawing object that it no longer requires. Before removing the object, the application should select it out of the device context. To do this, the application can select a different object of the same type back into the device context; typically, this different object is the original object for the device context.

When the hdc parameter identifies a metafile device context, the SelectObject function does not return the handle of the previously selected object. When the device context is a metafile, calling SelectObject with the hgdiobj parameter set to a value returned by a previous call to SelectObject can cause unpredictable results. Because metafiles perform their own object cleanup, an application need not reselect default objects when recording a metafile.

Memory device contexts are the only device contexts into which an application can select a bitmap. A bitmap can be selected into only one memory device context at a time. The format of the bitmap must either be monochrome or be compatible with the given device; if it is not, SelectObject returns an error.

### Example

The following example creates a pen, uses the SelectObject function to select it into a device context, uses the pen to draw a rectangle, selects the previous pen back into the device context, and uses the DeleteObject function to remove the pen that was just created:

```
HPEN hpen, hpenOld;
```

```
hpen = CreatePen(PS_SOLID, 6, RGB(0, 0, 255));
hpenOld = SelectObject(hdc, hpen);
```

```
Rectangle(hdc, 10, 10, 100, 100);
```

```
SelectObject(hdc, hpenOld);
```

```
DeleteObject(hpen);
```

See Also

DeleteObject, SelectClipRgn, SelectPalette

## Changes

For Windows 3.1, the `SelectObject` function returns the same value whether or not it is used in a metafile. Under previous versions of Windows, the `SelectObject` function returned a nonzero value for success and zero for failure when it was used in a metafile.

### SetAbortProc (3.1)

```
int SetAbortProc(hdc, abrtprc)
HDC hdc;          /* handle of device context */
ABORTPROC abrtprc;
/* instance address of abort function
/
```

The SetAbortProc function sets the application-defined procedure that allows a print job to be canceled during spooling. This function replaces the SETABORTPROC printer escape for Windows version 3.1.

Parameter	Description
hdc	Identifies the device context for the print job.
abrtprc	Specifies the procedure-instance address of the callback function. The address must have been created by using the <u>MakeProcInstance</u> function. For more information about the callback function, see the description of the <u>AbortProc</u> callback function.

#### Returns

The return value is greater than zero if the function is successful. Otherwise, it is less than zero.

#### See Also

AbortDoc, AbortProc, Escape

## SetBitmapBits (2.x)

```
LONG SetBitmapBits(hbmp, cBits, lpvBits)
HBITMAP hbmp;          /* handle of bitmap */
DWORD cBits;          /* number of bytes in bitmap array */
/* address of array with bitmap bits */
const void FAR* lpvBits;
```

The SetBitmapBits function sets the bits of the given bitmap, to the specified bit values.

Parameter	Description
hbmp	Identifies the bitmap to be set.
cBits	Specifies the number of bytes pointed to by the lpvBits parameter.
lpvBits	Points to an array of bytes for the bitmap bits.

### Returns

The return value is the number of bytes used in setting the bitmap bits, if the function is successful. Otherwise, the return value is zero.

### See Also

[GetBitmapBits](#)

## SetBitmapDimension (2.x)

DWORD SetBitmapDimension(hbmp, nWidth, nHeight)

HBITMAP hbmp; /\* handle of bitmap \*/

int nWidth; /\* bitmap width \*/

int nHeight; /\*

bitmap height

/

\*

The SetBitmapDimension function assigns a width and height to a bitmap, in 0.1-millimeter units. The graphics device interface (GDI) does not use these values except to return them when an application calls the GetBitmapDimension function.

Parameter	Description
hbmp	Identifies the bitmap.
nWidth	Specifies the bitmap width, in 0.1-millimeter units.
nHeight	Specifies the bitmap height, in 0.1-millimeter units.

### Returns

The return value is the dimensions of the previous bitmap, in 0.1-millimeter units, if the function is successful. The low-order word contains the previous width; the high-order word contains the previous height.

### See Also

GetBitmapDimension

### SetBitmapDimensionEx (3.1)

```
BOOL SetBitmapDimensionEx(hbm, nX, nY, lpSize)
HBITMAP hbm;          /* handle of bitmap */
int nX;                /* bitmap width */
/
int nY;                /
* bitmap height
/
SIZE FAR* lpSize;
* address of structure for prev. dimensions
/
```

The SetBitmapDimensionEx function assigns the preferred size to a bitmap, in 0.1-millimeter units. The graphics device interface (GDI) does not use these values, except to return them when an application calls the GetBitmapDimensionEx function.

Parameter	Description
hbm	Identifies the bitmap.
nX	Specifies the width of the bitmap, in 0.1-millimeter units.
nY	Specifies the height of the bitmap, in 0.1-millimeter units.
lpSize	Points to a <u>SIZE</u> structure. The previous bitmap dimensions are placed in this structure. If lpSize is <u>NULL</u> , nothing is returned.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### See Also

GetBitmapDimensionEx, SIZE



## SetBkColor (2.x)

```
COLORREF SetBkColor(hdc, clrref)
HDC hdc; /* handle of device context */
COLORREF clrref; /* color specification */
/
```

The SetBkColor function sets the current background color to the specified color.

Parameter	Description
hdc	Identifies the device context.
clrref	Specifies the new background color.

### Returns

The return value is the RGB value of the previous background color, if the function is successful. The return value is 0x80000000 if an error occurs.

### Comments

If the background mode is OPAQUE, the system uses the background color to fill the gaps in styled lines, the gaps between hatched lines in brushes, and the background in character cells. The system also uses the background color when converting bitmaps between color and monochrome device contexts.

If the device cannot display the specified color, the system sets the background color to the nearest physical color.

### Example

The following example uses the GetBkColor function to determine whether the current background color is white. If it is, the SetBkColor function sets it to red.

```
DWORD dwBackColor;

dwBackColor = GetBkColor(hdc);
if (dwBackColor == RGB(255, 255, 255)) { /* if color is white */
    SetBkColor(hdc, RGB(255, 0, 0)); /* sets color to red */
    TextOut(hdc, 100, 200, "SetBkColor test.", 16);
}
```

### See Also

BitBlt, GetBkColor, GetBkMode, SetBkMode, StretchBlt, RGB

## SetBkMode (2.x)

```
int SetBkMode(hdc, fnBkMode)
HDC hdc;          /* handle of device context */
int fnBkMode;     /* background mode */
```

The SetBkMode function sets the specified background mode. The background mode defines whether the system removes existing background colors on the drawing surface before drawing text, hatched brushes, or any pen style that is not a solid line.

Parameter	Description						
hdc	Identifies the device context.						
fnBkMode	Specifies the background mode to be set. This parameter can be one of the following values:						
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>OPAQUE</u></td><td>Background is filled with the current background color before the text, hatched brush, or pen is drawn. This is the default background mode.</td></tr><tr><td><u>TRANSPARENT</u></td><td>Background is not changed before drawing.</td></tr></tbody></table>	Value	Meaning	<u>OPAQUE</u>	Background is filled with the current background color before the text, hatched brush, or pen is drawn. This is the default background mode.	<u>TRANSPARENT</u>	Background is not changed before drawing.
Value	Meaning						
<u>OPAQUE</u>	Background is filled with the current background color before the text, hatched brush, or pen is drawn. This is the default background mode.						
<u>TRANSPARENT</u>	Background is not changed before drawing.						

### Returns

The return value is the previous background mode, if the function is successful.

### Example

The following example determines the current background mode by calling the GetBkMode function. If the mode is OPAQUE, the SetBkMode function sets it to TRANSPARENT.

```
int nBackMode;

nBackMode = GetBkMode(hdc);
if (nBackMode == OPAQUE) {
    TextOut(hdc, 90, 100, "This background mode is OPAQUE.", 31);
    SetBkMode(hdc, TRANSPARENT);
}
```

### See Also

GetBkColor, GetBkMode, SetBkColor

## OPAQUE 2

Background is filled with the current background color before the text, hatched brush, or pen is drawn.  
This is the default background mode.

OPAQUE 2

TRANSPARENT 1

Background is not changed before drawing.

TRANSPARENT 1

### SetBoundsRect (3.1)

```
UINT SetBoundsRect(hdc, lprcBounds, flags)
HDC hdc; /* handle of device context */
const RECT FAR* lprcBounds;
/* address of structure for rectangle */
/
UINT flags;
/* specifies information to return */
/
```

The SetBoundsRect function controls the accumulation of bounding-rectangle information for the specified device context.

Parameter	Description								
hdc	Identifies the device context to accumulate bounding rectangles for.								
lprcBounds	Points to a <u>RECT</u> structure that is used to set the bounding rectangle. Rectangle dimensions are given in logical coordinates. This parameter can be NULL.								
flags	Specifies how the new rectangle will be combined with the accumulated rectangle. This parameter may be a combination of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>DCB_ACCUMULATE</td><td>Add the rectangle specified by the lprcBounds parameter to the bounding rectangle (using a rectangle union operation).</td></tr><tr><td>DCB_DISABLE</td><td>Turn off bounds accumulation.</td></tr><tr><td>DCB_ENABLE</td><td>Turn on bounds accumulation. (The default setting for bounds accumulation is disabled.)</td></tr></tbody></table>	Value	Meaning	DCB_ACCUMULATE	Add the rectangle specified by the lprcBounds parameter to the bounding rectangle (using a rectangle union operation).	DCB_DISABLE	Turn off bounds accumulation.	DCB_ENABLE	Turn on bounds accumulation. (The default setting for bounds accumulation is disabled.)
Value	Meaning								
DCB_ACCUMULATE	Add the rectangle specified by the lprcBounds parameter to the bounding rectangle (using a rectangle union operation).								
DCB_DISABLE	Turn off bounds accumulation.								
DCB_ENABLE	Turn on bounds accumulation. (The default setting for bounds accumulation is disabled.)								

#### Returns

The return value is the current state of the bounding rectangle, if the function is successful. Like the flags parameter, the return value can be a combination of DCB\_ values, as shown in the following list:

Value	Meaning
DCB_ACCUMULATE	The bounding rectangle is not empty. (This value will always be set.)
DCB_DISABLE	Bounds accumulation is off.
DCB_ENABLE	Bounds accumulation is on.

#### Comments

Windows can maintain a bounding rectangle for all drawing operations. This rectangle can be queried and reset by the application. The drawing bounds are useful for invalidating bitmap caches.

#### See Also

[GetBoundsRect](#)

## SetBrushOrg (2.x)

```
DWORD SetBrushOrg(hdc, nXOrg, nYOrg)
HDC hdc;          /* handle of device context */
int nXOrg;        /* x-coordinate of new origin */
int nYOrg;        /* y-coordinate of new origin */
/
```

The SetBrushOrg function specifies the origin that GDI will assign to the next brush an application selects into the specified device context.

Parameter	Description
hdc	Identifies the device context.
nXOrg	Specifies the x-coordinate, in device units, of the new origin. This value must be in the range 0 through 7.
nYOrg	Specifies the y-coordinate, in device units, of the new origin. This value must be in the range 0 through 7.

### Returns

The return value is the coordinates, in device units, of the previous origin, if the function is successful. The low-order word contains the x-coordinate; the high-order word contains the y-coordinate.

### Comments

The default coordinates for the brush origin are (0, 0).

To alter the origin of a brush, an application should call the UnrealizeObject function, specifying the handle of the brush for which the origin will be set; call SetBrushOrg; and then call the SelectObject function to select the brush into the device context.

The SetBrushOrg function should not be used with stock objects.

### Example

The following example uses the SetBrushOrg function to shift the brush origin vertically by 5 pixels:

```
HBRUSH hbr, hbrOld;
SetBkMode(hdc, TRANSPARENT);
hbr = CreateHatchBrush(HS_CROSS, RGB(0, 0, 0));

UnrealizeObject(hbr);
SetBrushOrg(hdc, 0, 0);
hbrOld = SelectObject(hdc, hbr);

Rectangle(hdc, 0, 0, 200, 200);

hbr = SelectObject(hdc, hbrOld); /* deselects hbr */
UnrealizeObject(hbr); /* resets origin next time hbr selected */
SetBrushOrg(hdc, 3, 5);
hbrOld = SelectObject(hdc, hbr); /* selects hbr again */

Rectangle(hdc, 0, 0, 200, 200);

SelectObject(hdc, hbrOld);
DeleteObject(hbr);

See Also
GetBrushOrg, SelectObject, UnrealizeObject, HIWORD, LOWORD
```



## Corrections

The function purpose statement was incorrect. SetBrushOrg does not alter the origin of the current brush in a device context; instead, it sets the origin for the next brush to be selected into the device context. The original purpose statement read as follows: "The SetBrushOrg function sets the origin of the current brush for the specified device context."

### SetDIBits (3.0)

```
int SetDIBits(hdc, hbmp, uStartScan, cScanLines, lpvBits, lpbmi, fuColorUse)
HDC hdc; /* handle of device context */
HBITMAP hbmp; /* handle of bitmap */
/
UINT uStartScan; /* starting scan line */
/
UINT cScanLines; /* number of scan lines */
/
const void FAR* lpvBits; /* address of array with bitmap bits */
/
BITMAPINFO FAR* lpbmi; /* address of structure with bitmap data */
/
UINT fuColorUse; /* type of color indices to use */
/
```

The SetDIBits function sets the bits of a bitmap to the values given in a device-independent bitmap (DIB) specification.

Parameter	Description
hdc	Identifies the device context.
hbmp	Identifies the bitmap to set the data in.
uStartScan	Specifies the zero-based scan number of the first scan line in the buffer pointed to by the lpvBits parameter.
cScanLines	Specifies the number of scan lines in the lpvBits buffer to copy into the bitmap identified by the hbmp parameter.
lpvBits	Points to the device-independent bitmap bits that are stored as an array of bytes. The format of the bitmap values depends on the biBitCount member of the <u>BITMAPINFOHEADER</u> structure, which is the first member of the <u>BITMAPINFO</u> structure pointed to by the lpbmi parameter.
lpbmi	Points to a <u>BITMAPINFO</u> structure that contains information about the device-independent bitmap.
fuColorUse	Specifies whether the bmiColors member of the <u>BITMAPINFO</u> structure contains explicit <u>RGB</u> values or indices into the currently <u>realized logical</u> palette. This parameter must be <u>one</u> of the following values:
Value	Meaning
DIB_PAL_COLORS	The color table consists of an array of 16-bit indices into the palette of the device context identified by the hdc parameter.
DIB_RGB_COLORS	The color table contains literal <u>RGB</u> values.

#### Returns

The return value is the number of scan lines copied, if the function is successful. Otherwise, it is zero.

#### Comments

The bitmap identified by the hbmp parameter must not be selected into a device context when the application calls this function.

To reduce the amount of memory required to set bits from a large device-independent bitmap on a device surface, an application can band the output by repeatedly calling the SetDIBitsToDevice function, placing a different portion of the entire bitmap into the lpvBits buffer each time. The values of the uStartScan and cScanLines parameters identify the portion of the entire bitmap that is contained in the lpvBits buffer.

The origin of a device-independent bitmap is the bottom-left corner of the bitmap, not the top-left corner, which is the origin when the mapping mode is MM\_TEXT. GDI performs the necessary transformation to display the image correctly.

See Also

SetDIBitsToDevice, BITMAPCOREINFO, BITMAPINFO, BITMAPINFOHEADER

### SetDIBitsToDevice (3.0)

```
int SetDIBitsToDevice(hdc, XDest, YDest, cx, cy, XSrc, YSrc, uStartScan, cScanLines, lpvBits, lpbmi,
                    fuColorUse)
HDC hdc; /* handle of device context */
int XDest; /* x-
coordinate origin of destination rect
/
int YDest;
* y-coordinate origin of destination rect
/
int cx;
* rectangle width
/
int cy;
* rectangle height
/
int XSrc;
* x-coordinate origin of source rect
/
int YSrc;
* y-coordinate origin of source rect
/
UINT uStartScan;
* number of first scan line in array
/
UINT cScanLines;
* number of scan lines
/
void FAR* lpvBits;
* address of array with DIB bits
/
BITMAPINFO FAR* lpbmi;
* address of structure with bitmap info
/
UINT fuColorUse;
* RGB or palette indices
/
```

The SetDIBitsToDevice function sets bits from a device-independent bitmap (DIB) directly on a device surface. The device coordinates specified define a rectangle within the total bitmap. SetDIBitsToDevice sets the bits in this rectangle directly on the display surface of the output device associated with the given device context, at the specified logical coordinates.

Parameter	Description
hdc	Identifies the device context.
XDest	Specifies the logical x-coordinate of the origin of the destination rectangle.
YDest	Specifies the logical y-coordinate of the origin of the destination rectangle.
cx	Specifies the x-extent, in device units, of the rectangle in the bitmap.
cy	Specifies the y-extent, in device units, of the rectangle in the bitmap.
XSrc	Specifies the x-coordinate, in device units, of the source rectangle in the bitmap.
YSrc	Specifies the y-coordinate, in device units, of the source rectangle in the bitmap.
uStartScan	Specifies the scan-line number of the device-independent bitmap that is contained in the first scan line of the buffer pointed to by the lpvBits parameter.
cScanLines	Specifies the number of scan lines in the lpvBits buffer to copy to the device.
lpvBits	Points to the DIB bits that are stored as an array of bytes.
lpbmi	Points to a <u>BITMAPINFO</u> structure that contains information about the bitmap.
fuColorUse	Specifies whether the bmiColors member of the lpbmi parameter contains explicit <u>RGB</u> values or indices into the currently realized logical palette. This parameter must be <u>one</u> of the following values:

Value	Meaning
DIB_PAL_COLORS	The color table consists of an array of 16-bit indices into the currently realized logical palette.
DIB_RGB_COLORS	The color table contains literal <u>RGB</u> values.

#### Returns

The return value is the number of scan lines set, if the function is successful.

#### Comments

The origin of a device-independent bitmap is the bottom-left corner of the bitmap, not the top-left corner, which is the origin when the mapping mode is MM\_TEXT. GDI performs the necessary transformation to display the image correctly.

To reduce the amount of memory required to set bits from a large device-independent bitmap on a device surface, an application can band the output by repeatedly calling SetDIBitsToDevice, placing a different portion of the entire bitmap into the lpvBits buffer each time. The values of the uStartScan and cScanLines parameters identify the portion of the entire bitmap that is contained in the lpvBits buffer.

#### See Also

SetDIBits, BITMAPCOREINFO, BITMAPINFO

## SetMapMode (2.x)

```
int SetMapMode(hdc, fnMapMode)
HDC hdc;          /* handle of device context */
int fnMapMode;    /* mapping mode to set */
```

The `SetMapMode` function sets the mapping mode of the given device context. The mapping mode defines the unit of measure used to convert logical units to device units; it also defines the orientation of the device's x- and y-axes. GDI uses the mapping mode to convert logical coordinates into the appropriate device coordinates.

Parameter	Description																		
hdc	Identifies the device context.																		
fnMapMode	Specifies the new mapping mode. This parameter can be any one of the following values:																		
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MM_ANISOTROPIC</td><td>Logical units are converted to arbitrary units with arbitrarily scaled axes. Setting the mapping mode to MM_ANISOTROPIC does not change the current window or viewport settings. To change the units, orientation, and scaling, an application should use the <u>SetWindowExt</u> and <u>SetViewportExt</u> functions.</td></tr><tr><td>MM_HIENGLISH</td><td>Each logical unit is converted to 0.001 inch. Positive x is to the right; positive y is up.</td></tr><tr><td>MM_HIMETRIC</td><td>Each logical unit is converted to 0.01 millimeter. Positive x is to the right; positive y is up.</td></tr><tr><td>MM_ISOTROPIC</td><td>Logical units are converted to arbitrary units with equally scaled axes; that is, one unit along the x-axis is equal to one unit along the y-axis. The <u>SetWindowExt</u> and <u>SetViewportExt</u> functions must be used to specify the <u>desired units</u> and the orientation of the axes. <u>GDI</u> makes adjustments as necessary to ensure that the x and y units remain the same size.</td></tr><tr><td>MM_LOENGLISH</td><td>Each logical unit is converted to 0.01 inch. Positive x is to the right; positive y is up.</td></tr><tr><td>MM_LOMETRIC</td><td>Each logical unit is converted to 0.1 millimeter. Positive x is to the right; positive y is up.</td></tr><tr><td>MM_TEXT</td><td>Each logical unit is converted to one device pixel. Positive x is to the right; positive y is down.</td></tr><tr><td>MM_TWIPS</td><td>Each logical unit is converted to 1/20 of a point. (Because a point is 1/72 inch, a twip is 1/1440 inch). Positive x is to the right; positive y is up.</td></tr></tbody></table>	Value	Meaning	MM_ANISOTROPIC	Logical units are converted to arbitrary units with arbitrarily scaled axes. Setting the mapping mode to MM_ANISOTROPIC does not change the current window or viewport settings. To change the units, orientation, and scaling, an application should use the <u>SetWindowExt</u> and <u>SetViewportExt</u> functions.	MM_HIENGLISH	Each logical unit is converted to 0.001 inch. Positive x is to the right; positive y is up.	MM_HIMETRIC	Each logical unit is converted to 0.01 millimeter. Positive x is to the right; positive y is up.	MM_ISOTROPIC	Logical units are converted to arbitrary units with equally scaled axes; that is, one unit along the x-axis is equal to one unit along the y-axis. The <u>SetWindowExt</u> and <u>SetViewportExt</u> functions must be used to specify the <u>desired units</u> and the orientation of the axes. <u>GDI</u> makes adjustments as necessary to ensure that the x and y units remain the same size.	MM_LOENGLISH	Each logical unit is converted to 0.01 inch. Positive x is to the right; positive y is up.	MM_LOMETRIC	Each logical unit is converted to 0.1 millimeter. Positive x is to the right; positive y is up.	MM_TEXT	Each logical unit is converted to one device pixel. Positive x is to the right; positive y is down.	MM_TWIPS	Each logical unit is converted to 1/20 of a point. (Because a point is 1/72 inch, a twip is 1/1440 inch). Positive x is to the right; positive y is up.
Value	Meaning																		
MM_ANISOTROPIC	Logical units are converted to arbitrary units with arbitrarily scaled axes. Setting the mapping mode to MM_ANISOTROPIC does not change the current window or viewport settings. To change the units, orientation, and scaling, an application should use the <u>SetWindowExt</u> and <u>SetViewportExt</u> functions.																		
MM_HIENGLISH	Each logical unit is converted to 0.001 inch. Positive x is to the right; positive y is up.																		
MM_HIMETRIC	Each logical unit is converted to 0.01 millimeter. Positive x is to the right; positive y is up.																		
MM_ISOTROPIC	Logical units are converted to arbitrary units with equally scaled axes; that is, one unit along the x-axis is equal to one unit along the y-axis. The <u>SetWindowExt</u> and <u>SetViewportExt</u> functions must be used to specify the <u>desired units</u> and the orientation of the axes. <u>GDI</u> makes adjustments as necessary to ensure that the x and y units remain the same size.																		
MM_LOENGLISH	Each logical unit is converted to 0.01 inch. Positive x is to the right; positive y is up.																		
MM_LOMETRIC	Each logical unit is converted to 0.1 millimeter. Positive x is to the right; positive y is up.																		
MM_TEXT	Each logical unit is converted to one device pixel. Positive x is to the right; positive y is down.																		
MM_TWIPS	Each logical unit is converted to 1/20 of a point. (Because a point is 1/72 inch, a twip is 1/1440 inch). Positive x is to the right; positive y is up.																		

### Returns

The return value is the previous mapping mode, if the function is successful.

### Comments

The MM\_TEXT mode allows applications to work in device pixels, where one unit is equal to one pixel. The physical size of a pixel varies from device to device.

The MM\_HIENGLISH, MM\_HIMETRIC, MM\_LOENGLISH, MM\_LOMETRIC, and MM\_TWIPS modes are useful for applications that must draw in physically meaningful units (such as inches or millimeters).

The MM\_ISOTROPIC mode ensures a 1:1 aspect ratio, which is useful when it is important to preserve the exact shape of an image.

The MM\_ANISOTROPIC mode allows the x- and y-coordinates to be adjusted independently.

### Example

The following example uses the `SetMapMode` function to set the mapping mode to MM\_TWIPS and then uses the CreateFont function to create an 18-point logical font:

```
HFONT hfont, hfontOld;
```

```
int MapModePrevious, iPtSize = 18;
PSTR pszFace = "MS Serif";

MapModePrevious = SetMapMode(hdc, MM_TWIPS);
hfont = CreateFont(-iPtSize * 20, 0, 0, 0, 0, /* specify pt size */
    0, 0, 0, 0, 0, 0, 0, 0, pszFace); /* and face name only */

hfontOld = SelectObject(hdc, hfont);

TextOut(hdc, 100, -500, pszFace, strlen(pszFace));
SetMapMode(hdc, MapModePrevious);
SelectObject(hdc, hfontOld);
DeleteObject(hfont);

See Also
GetMapMode, SetViewportExt, SetWindowExt
```

## SetMapperFlags (2.x)

```
DWORD SetMapperFlags(hdc, fdwMatch)
HDC hdc; /* handle of device context */
DWORD fdwMatch; /* mapper flag */
```

The SetMapperFlags function changes the method used by the font mapper when it converts a logical font to a physical font. An application can use SetMapperFlags to cause the font mapper to attempt to choose only a physical font that exactly matches the aspect ratio of the specified device.

Parameter	Description
hdc	Identifies a device context.
fdwMatch	Specifies whether the font mapper attempts to match a font's aspect height and width to the device. When this value is ASPECT_FILTERING, the mapper selects only fonts whose x-aspect and y-aspect exactly match those of the specified device, and the remaining bits are ignored.

### Returns

The return value is the previous value of the font-mapper flag, if the function is successful.

### Comments

An application that uses only raster fonts can use the SetMapperFlags function to ensure that the font selected by the font mapper is attractive and readable on the specified device. Applications that use scalable (TrueType) fonts typically do not use SetMapperFlags.

If no physical font has an aspect ratio that matches the specifications in the logical font, GDI chooses a new aspect ratio and selects a font that matches this new aspect ratio.



## SetMetaFileBits (2.x)

```
HGLOBAL SetMetaFileBits(hmf)
HMETAFILE hmf;          /* handle of metafile */
```

The SetMetaFileBits function creates a memory metafile from the data in the given global memory object.

Parameter	Description
hmf	Identifies the global memory object that contains the metafile data. The object must have been created by a previous call to the <u>GetMetaFileBits</u> function. Note that this global handle must be cast to an HMETAFILE type to avoid compiler warnings.

### Returns

The return value is the handle of a memory metafile, if the function is successful. Otherwise, it is NULL.

### Comments

After the SetMetaFileBits function returns, the metafile handle it returns must be used instead of the hmf handle to refer to the metafile. If SetMetaFileBits is successful, the application should not use or free the memory handle specified by the hmf parameter, because that handle is reused by Windows.

When the application no longer needs the metafile handle, it should free the handle by calling the DeleteMetaFile function.

### See Also

GetMetaFileBits, GlobalFree, SetMetaFileBitsBetter

## SetMetaFileBitsBetter (3.1)

```
HGLOBAL SetMetaFileBitsBetter(hmf)
HMETAFILE hmf;          /* handle of the metafile */
```

The SetMetaFileBitsBetter function creates a memory metafile from the data in the specified global-memory object.

Parameter	Description
hmf	Identifies the global-memory object that contains the metafile data. The object must have been created by a previous call to the <a href="#">GetMetaFileBits</a> function. Note that this global handle must be cast to an HMETAFILE type to avoid compiler warnings.

### Returns

The return value is the handle of a memory metafile, if the function is successful. Otherwise, the return value is NULL.

### Comments

The global-memory handle returned by SetMetaFileBitsBetter is owned by GDI, not by the application. This enables applications that use metafiles to support object linking and embedding (OLE) to use metafiles that persist beyond the termination of the application. An OLE application should always use SetMetaFileBitsBetter instead of the [SetMetaFileBits](#) function.

After the SetMetaFileBitsBetter function returns, the metafile handle returned by the function should be used to refer to the metafile, instead of the handle identified by the hmf parameter.

### See Also

[GetMetaFileBits](#), [SetMetaFileBits](#)

### SetPaletteEntries (3.0)

```
UINT SetPaletteEntries(hpal, iStart, cEntries, lppe)
HPALETTE hpal; /* handle of palette */
UINT iStart;
* index of first entry to set
/
UINT cEntries;
* number of entries to set
/
const PALETTEENTRY FAR* lppe;
* address of array of structures
/
```

The SetPaletteEntries function sets RGB color values and flags in a range of entries in the given logical palette.

Parameter	Description
hpal	Identifies the logical palette.
iStart	Specifies the first logical-palette entry to be set.
cEntries	Specifies the number of logical-palette entries to be set.
lppe	Points to the first member of an array of <u>PALETTEENTRY</u> structures containing the <u>RGB</u> values and flags.

#### Returns

The return value is the number of entries set in the logical palette, if the function is successful. Otherwise, it is zero.

#### Comments

If the logical palette is selected into a device context when the application calls the SetPaletteEntries function, the changes will not take effect until the application calls the RealizePalette function.

#### See Also

RealizePalette, PALETTEENTRY

## SetPixel (2.x)

```
COLORREF SetPixel(hdc, nXPos, nYPos, clrref)
HDC hdc; /* handle of device context */
int nXPos; /* x-coordinate of pixel to set */
set
/
int nYPos;
* y-coordinate of pixel to set
/
COLORREF clrref;
* color of set pixel
/
```

The SetPixel function sets the pixel at the specified coordinates to the closest approximation of the given color. The point must be in the clipping region; if it is not, the function does nothing.

Parameter	Description
hdc	Identifies the device context.
nXPos	Specifies the logical x-coordinate of the point to be set.
nYPos	Specifies the logical y-coordinate of the point to be set.
clrref	Specifies the color to be used to paint the point.

### Returns

The return value is the RGB value for the color the point is painted, if the function is successful. This value can be different from the specified value if an approximation of that color is used. The return value is -1 if the function fails (if the point is outside the clipping region).

### Comments

Not all devices support the SetPixel function. To discover whether a device supports raster operations, an application can call the GetDeviceCaps function using the RC\_BITBLT index.

### See Also

GetDeviceCaps, GetPixel

## SetPolyFillMode (2.x)

```
int SetPolyFillMode(hdc, fnMode)
HDC hdc;      /* handle of device context */
int fnMode;   /* polygon-filling mode */
```

The SetPolyFillMode function sets the specified polygon-filling mode.

Parameter	Description
hdc	Identifies the device context.
fnMode	Specifies the new filling mode. This value may be either ALTERNATE or WINDING. The default mode is ALTERNATE.

### Returns

The return value specifies the previous filling mode, if the function is successful. Otherwise, it is zero.

### Comments

When the polygon-filling mode is ALTERNATE, the system fills the area between odd-numbered and even-numbered polygon sides on each scan line. That is, the system fills the area between the first and second side, between the third and fourth side, and so on.

When the polygon-filling mode is WINDING, the system uses the direction in which a figure was drawn to determine whether to fill an area. Each line segment in a polygon is drawn in either a clockwise or a counterclockwise direction. Whenever an imaginary line drawn from an enclosed area to the outside of a figure passes through a clockwise line segment, a count is incremented (increased by one); when the line passes through a counterclockwise line segment, the count is decremented (decreased by one). The area is filled if the count is nonzero when the line reaches the outside of the figure.

### Example

The following example uses winding mode to draw the same figure twice. The figure is a rectangle that completely encloses a triangle. The first time the figure is drawn, both the rectangle and the triangle are drawn clockwise, and both the rectangle and the triangle are filled. The second time, the rectangle is drawn clockwise, but the triangle is drawn counterclockwise; the rectangle is filled, but the triangle is not. (If the figures had been drawn using alternate mode, the rectangle would have been filled and the triangle would not have been filled, in both cases.)

```
HBRUSH hbrGray, hbrPrevious;

/*
 * Define the points for a clockwise triangle in a clockwise
 * rectangle.
 */

POINT aPolyPoints[9] = {{ 50, 60 }, { 250, 60 }, { 250, 260 },
                        { 50, 260 }, { 50, 60 }, { 150, 80 },
                        { 230, 240 }, { 70, 240 }, { 150, 80 }};

int aPolyCount[] = { 5, 4 };
int cValues, i;

hbrGray = GetStockObject(GRAY_BRUSH);
hbrPrevious = SelectObject(hdc, hbrGray);

cValues = sizeof(aPolyCount) / sizeof(int);

SetPolyFillMode(hdc, WINDING); /* sets winding mode */
PolyPolygon(hdc, aPolyPoints, aPolyCount, cValues);

/* Define the triangle counter-clockwise */

aPolyPoints[6].x = 70; aPolyPoints[6].y = 240;
aPolyPoints[7].x = 230; aPolyPoints[7].y = 240;

for (i = 0; i < sizeof(aPolyPoints) / sizeof(POINT); i++)
    aPolyPoints[i].x += 300; /* moves figure 300 units right */
```

PolyPolygon(hdc, aPolyPoints, aPolyCount, cValues);

SelectObject(hdc, hbrPrevious);

See Also

GetPolyFillMode, PolyPolygon

## SetRectRgn (2.x)

```
void SetRectRgn(hrgn, nLeftRect, nTopRect, nRightRect, nBottomRect)
HRGN hrgn;          /* handle of region */
int nLeftRect;     /* x-
coordinate top-left corner of rectangle
/
int nTopRect;
* y-coordinate top-left corner of rectangle
/
int nRightRect;
* x-coordinate bottom-right corner of rectangle
/
int nBottomRect;
* y-coordinate bottom-right corner of rectangle
/
```

The SetRectRgn function changes the given region into a rectangular region with the specified coordinates.

Parameter	Description
hrgn	Identifies the region.
nLeftRect	Specifies the x-coordinate of the upper-left corner of the rectangular region.
nTopRect	Specifies the y-coordinate of the upper-left corner of the rectangular region.
nRightRect	Specifies the x-coordinate of the lower-right corner of the rectangular region.
nBottomRect	Specifies the y-coordinate of the lower-right corner of the rectangular region.

### Returns

This function does not return a value.

### Comments

Applications can use this function instead of the CreateRectRgn function to avoid allocating more memory from the GDI heap. Because the memory allocated for the hrgn parameter is reused, no new allocation is performed.

### Example

The following example uses the CreateRectRgn function to create a rectangular region and then calls the SetRectRgn function to change the region coordinates:

```
HRGN hrgn;

hrgn = CreateRectRgn(10, 10, 30, 30);
PaintRgn(hdc, hrgn);

SetRectRgn(hrgn, 50, 50, 150, 200);
PaintRgn(hdc, hrgn);

DeleteObject(hrgn);
```

### See Also

CreateRectRgn

## SetROP2 (2.x)

```
int SetROP2(hdc, fnDrawMode)
HDC hdc;          /* handle of device context */
int fnDrawMode;  /* new drawing mode */
/
```

The SetROP2 function sets the current drawing mode. The drawing mode specifies how the colors of the pen and the interior of filled objects are combined with the color already on the screen surface.

Parameter	Description																																		
hdc	Identifies the device context.																																		
fnDrawMode	Specifies the new drawing mode. This parameter can be one of the following values:																																		
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>R2_BLACK</td><td>Pixel is always black.</td></tr><tr><td>R2_WHITE</td><td>Pixel is always white.</td></tr><tr><td>R2_NOP</td><td>Pixel remains unchanged.</td></tr><tr><td>R2_NOT</td><td>Pixel is the inverse of the screen color.</td></tr><tr><td>R2_COPYPEN</td><td>Pixel is the pen color.</td></tr><tr><td>R2_NOTCOPYPEN</td><td>Pixel is the inverse of the pen color.</td></tr><tr><td>R2_MERGEPENNOT</td><td>Pixel is a combination of the pen color and the inverse of the screen color (final pixel = (~screen pixel)   pen).</td></tr><tr><td>R2_MASKPENNOT</td><td>Pixel is a combination of the colors common to both the pen and the inverse of the screen (final pixel = (~screen pixel) &amp; pen).</td></tr><tr><td>R2_MERGENOTPEN</td><td>Pixel is a combination of the screen color and the inverse of the pen color (final pixel = (~pen)   screen pixel).</td></tr><tr><td>R2_MASKNOTPEN</td><td>Pixel is a combination of the colors common to both the screen and the inverse of the pen (final pixel = (~pen) &amp; screen pixel).</td></tr><tr><td>R2_MERGEPEN</td><td>Pixel is a combination of the pen color and the screen color (final pixel = pen   screen pixel).</td></tr><tr><td>R2_NOTMERGEPEN</td><td>Pixel is the inverse of the R2_MERGEPEN color (final pixel = ~(pen   screen pixel)).</td></tr><tr><td>R2_MASKPEN</td><td>Pixel is a combination of the colors common to both the pen and the screen (final pixel = pen &amp; screen pixel).</td></tr><tr><td>R2_NOTMASKPEN</td><td>Pixel is the inverse of the R2_MASKPEN color (final pixel = ~(pen &amp; screen pixel)).</td></tr><tr><td>R2_XORPEN</td><td>Pixel is a combination of the colors that are in the pen and in the screen, but not in both (final pixel = pen ^ screen pixel).</td></tr><tr><td>R2_NOTXORPEN</td><td>Pixel is the inverse of the R2_XORPEN color (final pixel = ~(pen ^ screen pixel)).</td></tr></tbody></table>	Value	Meaning	R2_BLACK	Pixel is always black.	R2_WHITE	Pixel is always white.	R2_NOP	Pixel remains unchanged.	R2_NOT	Pixel is the inverse of the screen color.	R2_COPYPEN	Pixel is the pen color.	R2_NOTCOPYPEN	Pixel is the inverse of the pen color.	R2_MERGEPENNOT	Pixel is a combination of the pen color and the inverse of the screen color (final pixel = (~screen pixel)   pen).	R2_MASKPENNOT	Pixel is a combination of the colors common to both the pen and the inverse of the screen (final pixel = (~screen pixel) & pen).	R2_MERGENOTPEN	Pixel is a combination of the screen color and the inverse of the pen color (final pixel = (~pen)   screen pixel).	R2_MASKNOTPEN	Pixel is a combination of the colors common to both the screen and the inverse of the pen (final pixel = (~pen) & screen pixel).	R2_MERGEPEN	Pixel is a combination of the pen color and the screen color (final pixel = pen   screen pixel).	R2_NOTMERGEPEN	Pixel is the inverse of the R2_MERGEPEN color (final pixel = ~(pen   screen pixel)).	R2_MASKPEN	Pixel is a combination of the colors common to both the pen and the screen (final pixel = pen & screen pixel).	R2_NOTMASKPEN	Pixel is the inverse of the R2_MASKPEN color (final pixel = ~(pen & screen pixel)).	R2_XORPEN	Pixel is a combination of the colors that are in the pen and in the screen, but not in both (final pixel = pen ^ screen pixel).	R2_NOTXORPEN	Pixel is the inverse of the R2_XORPEN color (final pixel = ~(pen ^ screen pixel)).
Value	Meaning																																		
R2_BLACK	Pixel is always black.																																		
R2_WHITE	Pixel is always white.																																		
R2_NOP	Pixel remains unchanged.																																		
R2_NOT	Pixel is the inverse of the screen color.																																		
R2_COPYPEN	Pixel is the pen color.																																		
R2_NOTCOPYPEN	Pixel is the inverse of the pen color.																																		
R2_MERGEPENNOT	Pixel is a combination of the pen color and the inverse of the screen color (final pixel = (~screen pixel)   pen).																																		
R2_MASKPENNOT	Pixel is a combination of the colors common to both the pen and the inverse of the screen (final pixel = (~screen pixel) & pen).																																		
R2_MERGENOTPEN	Pixel is a combination of the screen color and the inverse of the pen color (final pixel = (~pen)   screen pixel).																																		
R2_MASKNOTPEN	Pixel is a combination of the colors common to both the screen and the inverse of the pen (final pixel = (~pen) & screen pixel).																																		
R2_MERGEPEN	Pixel is a combination of the pen color and the screen color (final pixel = pen   screen pixel).																																		
R2_NOTMERGEPEN	Pixel is the inverse of the R2_MERGEPEN color (final pixel = ~(pen   screen pixel)).																																		
R2_MASKPEN	Pixel is a combination of the colors common to both the pen and the screen (final pixel = pen & screen pixel).																																		
R2_NOTMASKPEN	Pixel is the inverse of the R2_MASKPEN color (final pixel = ~(pen & screen pixel)).																																		
R2_XORPEN	Pixel is a combination of the colors that are in the pen and in the screen, but not in both (final pixel = pen ^ screen pixel).																																		
R2_NOTXORPEN	Pixel is the inverse of the R2_XORPEN color (final pixel = ~(pen ^ screen pixel)).																																		

### Returns

The return value specifies the previous drawing mode, if the function is successful.

### Comments

The drawing mode is for raster devices only; it does not apply to vector devices.

Drawing modes are binary raster-operation codes representing all possible Boolean combinations of two variables. These values are created by using the binary operations AND, OR, and XOR (exclusive OR) and the unary operation NOT.

### See Also

[GetDeviceCaps](#), [GetROP2](#)



## SetStretchBltMode (2.x)

```
int SetStretchBltMode(hdc, fnStretchMode)
HDC hdc; /* handle of device context */
int fnStretchMode; /* bitmap-stretching mode */
mode
/
```

The SetStretchBltMode function sets the bitmap-stretching mode. The bitmap-stretching mode defines how information is removed from bitmaps that are compressed by using the [StretchBlt](#) function.

Parameter	Description
hdc	Identifies the device context.
fnStretchMode	Specifies the new bitmap-stretching mode. This parameter can be one of the following values:

Value	Meaning
<a href="#">STRETCH_ANDSCANS</a>	Uses the AND operator to combine eliminated lines with the remaining lines. This mode preserves black pixels at the expense of colored or white pixels. It is the default mode.
<a href="#">STRETCH_DELETESCANS</a>	Deletes the eliminated lines. Information in the eliminated lines is not preserved.
<a href="#">STRETCH_ORSCANS</a>	Uses the OR operator to combine eliminated lines with the remaining lines. This mode preserves colored or white pixels at the expense of black pixels.

### Returns

The return value is the previous stretching mode, if the function is successful. It can be [STRETCH\\_ANDSCANS](#), [STRETCH\\_DELETESCANS](#), or [STRETCH\\_ORSCANS](#).

### Comments

The [STRETCH\\_ANDSCANS](#) and [STRETCH\\_ORSCANS](#) modes are typically used to preserve foreground pixels in monochrome bitmaps. The [STRETCH\\_DELETESCANS](#) mode is typically used to preserve color in color bitmaps.

### See Also

[GetStretchBltMode](#), [StretchBlt](#), [StretchDIBits](#)

## STRETCH\_ANDSCANS 1

Uses the AND operator to combine eliminated lines with the remaining lines. This mode preserves black pixels at the expense of colored or white pixels. It is the default mode.

STRETCH\_ANDSCANS 1

STRETCH\_DELETESCANS 3

Deletes the eliminated lines. Information in the eliminated lines is not preserved.

STRETCH\_DELETESCANS 3

## STRETCH\_ORSCANS 2

Uses the OR operator to combine eliminated lines with the remaining lines. This mode preserves colored or white pixels at the expense of black pixels.

STRETCH\_ORSCANS 2

## SetSystemPaletteUse (3.0)

```
UINT SetSystemPaletteUse(hdc, fuStatic)
HDC hdc;          /* handle of device context */
UINT fuStatic;    /* system-palette contents */
```

The `SetSystemPaletteUse` function sets the use of static colors in the system palette. The default system palette contains 20 static colors, which are not changed when an application realizes its logical palette. An application can use `SetSystemPaletteUse` to change this to two static colors (black and white).

Parameter	Description						
<code>hdc</code>	Identifies the device context. This device context must support color palettes.						
<code>fuStatic</code>	Specifies the new use of the system palette. This parameter can be either of the following values:						
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><code>SYSPAL_NOSTATIC</code></td><td>System palette contains no static colors except black and white.</td></tr><tr><td><code>SYSPAL_STATIC</code></td><td>System palette contains static colors that will not change when an application realizes its logical palette.</td></tr></tbody></table>	Value	Meaning	<code>SYSPAL_NOSTATIC</code>	System palette contains no static colors except black and white.	<code>SYSPAL_STATIC</code>	System palette contains static colors that will not change when an application realizes its logical palette.
Value	Meaning						
<code>SYSPAL_NOSTATIC</code>	System palette contains no static colors except black and white.						
<code>SYSPAL_STATIC</code>	System palette contains static colors that will not change when an application realizes its logical palette.						

### Returns

The return value is the previous setting for the static colors in the system palette, if the function is successful. This setting is either `SYSPAL_NOSTATIC` or `SYSPAL_STATIC`.

### Comments

An application must call this function only when its window is maximized and has the input focus.

If an application calls `SetSystemPaletteUse` with `fuStatic` set to `SYSPAL_NOSTATIC`, Windows continues to set aside two entries in the system palette for pure white and pure black, respectively.

After calling this function with `fuStatic` set to `SYSPAL_NOSTATIC`, an application must follow these steps:

- 1 Call the `UnrealizeObject` function to force the graphics device interface (GDI) to remap the logical palette completely when it is realized.
- 2 Realize the logical palette.
- 3 Call the `GetSysColor` function to save the current system-color settings.
- 4 Call the `SetSysColors` function to set the system colors to reasonable values using black and white. For example, adjacent or overlapping items (such as window frames and borders) should be set to black and white, respectively.
- 5 Send the `WM_SYSCOLORCHANGE` message to other top-level windows to allow them to be redrawn with the new system colors.

When the application's window loses focus or closes, the application must perform the following steps:

- 1 Call `SetSystemPaletteUse` with the `fuStatic` parameter set to `SYSPAL_STATIC`.
- 2 Call `UnrealizeObject` to force GDI to remap the logical palette completely when it is realized.
- 3 Realize the logical palette.
- 4 Restore the system colors to their previous values.
- 5 Send the `WM_SYSCOLORCHANGE` message.

### See Also

[GetSysColor](#), [SetSysColors](#), [SetSystemPaletteUse](#), [UnrealizeObject](#)



## SetTextAlign (2.x)

```
UINT SetTextAlign(hdc, fuAlign)
HDC hdc;          /* handle of device context */
UINT fuAlign;     /* text-alignment flags */
```

The SetTextAlign function sets the text-alignment flags for the given device context.

Parameter	Description																						
hdc	Identifies the device context.																						
fuAlign	Specifies text-alignment flags. The flags specify the relationship between a point and a rectangle that bounds the text. The point can be either the current position or coordinates specified by a text-output function (such as the <u>ExtTextOut</u> function). The rectangle that bounds the text is defined by the adjacent character cells in the text string. The fuAlign parameter can be one or more flags from the following three categories. Choose only one flag from each category. The first category affects text alignment in the x-direction: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>TA_CENTER</td><td>Aligns the point with the horizontal center of the bounding rectangle.</td></tr><tr><td>TA_LEFT</td><td>Aligns the point with the left side of the bounding rectangle. This is the default setting.</td></tr><tr><td>TA_RIGHT</td><td>Aligns the point with the right side of the bounding rectangle.</td></tr></tbody></table> The second category affects text alignment in the y-direction: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>TA_BASELINE</td><td>Aligns the point with the base line of the chosen font.</td></tr><tr><td>TA_BOTTOM</td><td>Aligns the point with the bottom of the bounding rectangle.</td></tr><tr><td>TA_TOP</td><td>Aligns the point with the top of the bounding rectangle. This is the default setting.</td></tr></tbody></table> The third category determines whether the current position is updated when text is written: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>TA_NOUPDATECP</td><td>Does not update the current position after each call to a text-output function. This is the default setting.</td></tr><tr><td>TA_UPDATECP</td><td>Updates the current x-position after each call to a text-output function. The new position is at the right side of the bounding rectangle for the text. When this flag is set, the coordinates specified in calls to the <u>TextOut</u> function are ignored.</td></tr></tbody></table>	Value	Meaning	TA_CENTER	Aligns the point with the horizontal center of the bounding rectangle.	TA_LEFT	Aligns the point with the left side of the bounding rectangle. This is the default setting.	TA_RIGHT	Aligns the point with the right side of the bounding rectangle.	Value	Meaning	TA_BASELINE	Aligns the point with the base line of the chosen font.	TA_BOTTOM	Aligns the point with the bottom of the bounding rectangle.	TA_TOP	Aligns the point with the top of the bounding rectangle. This is the default setting.	Value	Meaning	TA_NOUPDATECP	Does not update the current position after each call to a text-output function. This is the default setting.	TA_UPDATECP	Updates the current x-position after each call to a text-output function. The new position is at the right side of the bounding rectangle for the text. When this flag is set, the coordinates specified in calls to the <u>TextOut</u> function are ignored.
Value	Meaning																						
TA_CENTER	Aligns the point with the horizontal center of the bounding rectangle.																						
TA_LEFT	Aligns the point with the left side of the bounding rectangle. This is the default setting.																						
TA_RIGHT	Aligns the point with the right side of the bounding rectangle.																						
Value	Meaning																						
TA_BASELINE	Aligns the point with the base line of the chosen font.																						
TA_BOTTOM	Aligns the point with the bottom of the bounding rectangle.																						
TA_TOP	Aligns the point with the top of the bounding rectangle. This is the default setting.																						
Value	Meaning																						
TA_NOUPDATECP	Does not update the current position after each call to a text-output function. This is the default setting.																						
TA_UPDATECP	Updates the current x-position after each call to a text-output function. The new position is at the right side of the bounding rectangle for the text. When this flag is set, the coordinates specified in calls to the <u>TextOut</u> function are ignored.																						

### Returns

The return value is the previous text-alignment settings, if the function is successful. The low-order byte contains the horizontal setting; the high-order byte contains the vertical setting. Otherwise, the return value is zero.

### Comments

The text-alignment flags set by SetTextAlign are used by the TextOut and ExtTextOut functions.

### Example

The following example uses the GetTextFace function to retrieve the name of the current typeface, calls SetTextAlign so that the current position is updated when the TextOut function is called, and then writes some introductory text and the name of the typeface by calling TextOut:

```
int nFaceNameLen;
char aFaceName[ 80 ];

nFaceNameLen = GetTextFace(hdc, /* returns length of string */
    sizeof(aFaceName), /* size of face-name buffer */
    (LPSTR) aFaceName); /* address of face-name buffer */
```

```
SetTextAlign(hdc,  
    TA_UPDATECP); /* updates current position */  
MoveTo(hdc, 100, 100); /* sets current position*/  
TextOut(hdc, 0, 0, /* uses current position for text */  
    "This is the current face name: ", 31);  
TextOut(hdc, 0, 0, aFaceName, nFaceNameLen);
```

See Also

ExtTextOut, GetTextAlign, TextOut

## SetTextCharacterExtra (2.x)

```
int SetTextCharacterExtra(hdc, nExtraSpace)
HDC hdc;          /* handle of device context */
int nExtraSpace; /* extra character */
spacing          *
/
```

The `SetTextCharacterExtra` function sets the amount of intercharacter spacing. The graphics device interface (GDI) adds this spacing to each character, including break characters, when it writes a line of text to the device context.

Parameter	Description
<code>hdc</code>	Identifies the device context.
<code>nExtraSpace</code>	Specifies the amount of extra space, in logical units, to be added to each character. If the current mapping mode is not <code>MM_TEXT</code> , this parameter is transformed and rounded to the nearest pixel.

### Returns

The return value is the previous intercharacter spacing, if the function is successful.

### Comments

The default value for the amount of intercharacter spacing is zero.

### See Also

[GetTextCharacterExtra](#)

## SetTextColor (2.x)

```
COLORREF SetTextColor(hdc, clrref)
HDC hdc;          /* handle of device context */
COLORREF clrref; /* new color for text */
/
```

The SetTextColor function sets the text color to the specified color. The system uses the text color when writing text to a device context and also when converting bitmaps between color and monochrome device contexts.

Parameter	Description
hdc	Identifies the device context.
clrref	Specifies the color of the text.

### Returns

The return value is the RGB (red-green-blue) value for the previous text color, if the function is successful.

### Comments

If the device cannot represent the specified color, the system sets the text color to the nearest physical color.

The background color for a character is specified by the SetBkColor and SetBkMode functions.

### Example

The following example sets the text color to red if the GetTextColor function determines that the current text color is black. The text color is specified by using the RGB macro.

```
DWORD dwColor;
```

```
dwColor = GetTextColor(hdc);
if (dwColor == RGB(0, 0, 0)) /* if current color is black */
    SetTextColor(hdc, RGB(255, 0, 0)); /* sets color to red */
```

### See Also

GetTextColor, BitBlt, SetBkColor, SetBkMode, RGB

## SetTextJustification (2.x)

```
int SetTextJustification(hdc, nExtraSpace, cBreakChars)
HDC hdc;          /* handle of device context */
int nExtraSpace; /* space
to add to string
/
int cBreakChars;
* number of break characters in the string
/
```

The `SetTextJustification` function adds space to the break characters in a string. An application can use the `GetTextMetrics` function to retrieve a font's break character.

Parameter	Description
<code>hdc</code>	Identifies the device context.
<code>nExtraSpace</code>	Specifies the total extra space, in logical units, to be added to the line of text. If the current mapping mode is not <code>MM_TEXT</code> , the value given by this parameter is converted to the current mapping mode and rounded to the nearest device unit.
<code>cBreakChars</code>	Specifies the number of break characters in the line.

### Returns

The return value is 1 if the function is successful. Otherwise, it is zero.

### Comments

After the `SetTextJustification` function is called, a call to a text-output function (for example, `TextOut`) distributes the specified extra space evenly among the specified number of break characters. The break character is usually the space character (ASCII 32), but it may be defined by a font as some other character.

The `GetTextExtent` function is typically used with `SetTextJustification`. The `GetTextExtent` function computes the width of a given line before alignment. An application can determine how much space to specify in the `nExtraSpace` parameter by subtracting the value returned by `GetTextExtent` from the width of the string after alignment.

The `SetTextJustification` function can be used to align a line that contains multiple runs in different fonts. In this case, the line must be created piecemeal by aligning and writing each run separately.

Because rounding errors can occur during alignment, the system keeps a running error term that defines the current error. When aligning a line that contains multiple runs, `GetTextExtent` automatically uses this error term when it computes the extent of the next run, allowing the text-output function to blend the error into the new run. After each line has been aligned, this error term must be cleared to prevent it from being incorporated into the next line. The term can be cleared by calling `SetTextJustification` with the `nExtraSpace` parameter set to zero.

### Example

The following example writes two lines of text inside a box; one of the lines is aligned, and the other is not. The `GetTextExtent` function determines the width of the unaligned string. The `GetTextMetrics` function determines the break character that is used by the current font; this information is then used to determine how many break characters the string contains. The `SetTextJustification` function specifies the total amount of extra space and the number of break characters to distribute it among. After writing a line of aligned text, `SetTextJustification` is called again, to set the error term to zero.

```
POINT aPoints[5];
int iLMargin = 10, iRMargin = 10, iBoxWidth;
int cchString;
LPSTR lpszJustified = "Text to be justified in this test.";
DWORD dwExtent;
WORD wTextWidth;
TEXTMETRIC tm;
int j, cBreakChars;

aPoints[0].x = 100; aPoints[0].y = 50;
aPoints[1].x = 600; aPoints[1].y = 50;
aPoints[2].x = 600; aPoints[2].y = 200;
aPoints[3].x = 100; aPoints[3].y = 200;
```

```

aPoints[4].x = 100; aPoints[4].y = 50;

Polyline(hdc, aPoints, sizeof(aPoints) / sizeof(POINT));

TextOut(hdc, 100 + iLMargin, 100, "Unjustified text.", 17);
cchString = lstrlen(lpszJustified);
dwExtent = GetTextExtent(hdc, lpszJustified, cchString);
wTextWidth = LOWORD(dwExtent);

iBoxWidth = aPoints[1].x - aPoints[0].x;
GetTextMetrics(hdc, &tm);

for (cBreakChars = 0, j = 0; j < cchString; j++)
    if (*(lpszJustified + j) == (char) tm.tmBreakChar)
        cBreakChars++;

SetTextJustification(hdc,
    iBoxWidth - wTextWidth - (iLMargin + iRMargin),
    cBreakChars);

TextOut(hdc, 100 + iLMargin, 150, lpszJustified, cchString);

SetTextJustification(hdc, 0, 0); /* clears error term */

See Also
GetMapMode, GetTextExtent, GetTextMetrics, SetMapMode, TextOut

```

## SetViewportExt (2.x)

```
DWORD SetViewportExt(hdc, nXExtent, nYExtent)
HDC hdc;          /* handle of device context */
int nXExtent;     /* x-extent of viewport */
int nYExtent;     /* y-extent of viewport */
/
```

The SetViewportExt function sets the x- and y-extents of the viewport of the given device context. The viewport, along with the window, defines how points are converted from logical coordinates to device coordinates.

Parameter	Description
hdc	Identifies the device context.
nXExtent	Specifies the x-extent, in device units, of the viewport.
nYExtent	Specifies the y-extent, in device units, of the viewport.

### Returns

The return value is the previous viewport extents, in device units, if the function is successful. The low-order word contains the previous x-extent; the high-order word contains the previous y-extent. Otherwise, the return value is zero.

### Comments

When the following mapping modes are set, calls to the SetWindowExt and SetViewportExt functions are ignored:

MM\_HIENGLISH  
MM\_HIMETRIC  
MM\_LOENGLISH  
MM\_LOMETRIC  
MM\_TEXT  
MM\_TWIPS

When the mapping mode is MM\_ISOTROPIC, an application must call the SetWindowExt function before calling SetViewportExt.

The x- and y-extents of the viewport define how much the graphics device interface (GDI) must stretch or compress units in the logical coordinate system to fit units in the device coordinate system. For example, if the x-extent of the window is 2 and the x-extent of the viewport is 4, GDI converts two logical units (measured from the x-axis) into four device units. Similarly, if the y-extent of the window is 2 and the y-extent of the viewport is -1, GDI converts two logical units (measured from the y-axis) into one device unit.

The extents also define the relative orientation of the x- and y-axes in both coordinate systems. If the signs of matching window and viewport extents are the same, the axes have the same orientation. If the signs are different, the orientation is reversed. For example, if the y-extent of the window is 2 and the y-extent of the viewport is -1, GDI converts the positive y-axis in the logical coordinate system to the negative y-axis in the device coordinate system. If the x-extents are 2 and 4, GDI converts the positive x-axis in the logical coordinate system to the positive x-axis in the device coordinate system.

### Example

The following example uses the SetMapMode, SetWindowExt, and SetViewportExt functions to create a client area that is 10 logical units wide and 10 logical units high, and then draws a rectangle that is 4 logical units wide and 4 logical units high:

```
HDC hdc;
RECT rc;

GetClientRect(hwnd, &rc);
hdc = GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 10, 10);
SetViewportExt(hdc, rc.right, rc.bottom);
Rectangle(hdc, 3, 3, 7, 7);
ReleaseDC(hwnd, hdc);
```

See Also

[GetViewportExt](#), [SetViewportExtEx](#), [SetWindowExt](#)



### SetViewportExtEx (3.1)

```
BOOL SetViewportExtEx(hdc, nX, nY, lpSize)
HDC hdc; /* handle of device context */
int nX; /* x-extent of viewport */
int nY; /* y-extent of viewport */
SIZE FAR* lpSize;
/* address of struct. with prev. extents */
```

The SetViewportExtEx function sets the x- and y-extents of the viewport of the specified device context. The viewport, along with the window, defines how points are mapped from logical coordinates to device coordinates.

Parameter	Description
hdc	Identifies the device context.
nX	Specifies the x-extent of the viewport, in device units.
nY	Specifies the y-extent of the viewport, in device units.
lpSize	Points to a SIZE structure. The previous extents of the viewport, in device units, are placed in this structure. If lpSize is NULL, nothing is returned.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

When the following mapping modes are set, calls to the SetWindowExtEx and SetViewportExtEx functions are ignored:

MM\_HIENGLISH  
MM\_HIMETRIC  
MM\_LOENGLISH  
MM\_LOMETRIC  
MM\_TEXT  
MM\_TWIPS

When MM\_ISOTROPIC mode is set, an application must call the SetWindowExtEx function before it calls SetViewportExtEx.

#### See Also

SetViewportExt, SetWindowExtEx

## SetViewportOrg (2.x)

```
DWORD SetViewportOrg(hdc, nXOrigin, nYOrigin)
HDC hdc;          /* handle of device context */
int nXOrigin;     /* x-coordinate of new origin */
/
int nYOrigin;
* y-coordinate of new origin
/
```

The SetViewportOrg function sets the viewport origin of the specified device context. The viewport, along with the window, defines how points are converted from logical coordinates to device coordinates.

Parameter	Description
hdc	Identifies the device context.
nXOrigin	Specifies the x-coordinate, in device coordinates, of the origin of the viewport. This value must be within the range of the device coordinate system.
nYOrigin	Specifies the y-coordinate, in device coordinates, of the origin of the viewport. This value must be within the range of the device coordinate system.

### Returns

The return value is the coordinates of the previous viewport origin, in device units, if the function is successful. The low-order word contains the previous x-coordinate; the high-order word contains the previous y-coordinate. Otherwise, the return value is zero.

### Comments

The viewport origin is the origin of the device coordinate system. The graphics device interface (GDI) converts points from the logical coordinate system to device coordinates. (An application can specify the origin of the logical coordinate system by using the SetWindowOrg function.) GDI converts all points in the logical coordinate system to device coordinates in the same way as it converts the origin.

### Example

The following example uses the SetViewportOrg function to set the viewport origin to the center of the client area and then draws a rectangle centered over the origin:

```
HDC hdc;
RECT rc;

GetClientRect(hwnd, &rc);
hdc = GetDC(hwnd);
SetViewportOrg(hdc, rc.right/2, rc.bottom/2);
Rectangle(hdc, -100, -100, 100, 100);
ReleaseDC(hwnd, hdc);
```

### See Also

SetViewportOrgEx, SetWindowOrg

### SetViewportOrgEx (3.1)

```
BOOL SetViewportOrgEx(hdc, nX, nY, lpPoint)
HDC hdc;          /* handle of device context */
int nX;          /* x-coordinate
of new origin */
int nY;          /* y-coordinate of new origin */
POINT FAR* lpPoint;
/* address of struct. with prev. origin */
```

The SetViewportOrgEx function sets the viewport origin of the specified device context. The viewport, along with the window, defines how points are mapped from logical coordinates to device coordinates.

Parameter	Description
hdc	Identifies the device context.
nX	Specifies the x-coordinate, in device units, of the origin of the viewport.
nY	Specifies the y-coordinate, in device units, of the origin of the viewport.
lpPoint	Points to a <u>POINT</u> structure. The previous origin of the viewport, in device coordinates, is placed in <u>this structure</u> . If lpPoint is NULL, nothing is returned.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### See Also

SetViewportOrg, SetWindowOrgEx

## SetWindowExt (2.x)

```
DWORD SetWindowExt(hdc, nXExtent, nYExtent)
HDC hdc;          /* handle of device context */
int nXExtent;     /* x-extent of window */
int nYExtent;     /* y-extent of window */
/
```

The SetWindowExt function sets the x- and y-extents of the window associated with the given device context. The window, along with the viewport, defines how logical coordinates are converted to device coordinates.

Parameter	Description
hdc	Identifies the device context.
nXExtent	Specifies the x-extent, in logical units, of the window.
nYExtent	Specifies the y-extent, in logical units, of the window.

### Returns

The return value is the window's previous extents, in logical units, if the function is successful. The low-order word contains the previous x-extent; the high-order word contains the previous y-extent. Otherwise, the return value is zero.

### Comments

When the following mapping modes are set, calls to the SetWindowExt and SetViewportExt functions are ignored:

MM\_HIENGLISH  
MM\_HIMETRIC  
MM\_LOENGLISH  
MM\_LOMETRIC  
MM\_TEXT  
MM\_TWIPS

When MM\_ISOTROPIC mode is set, an application must call the SetWindowExt function before calling SetViewportExt.

The x- and y-extents of the window define how much the graphics device interface (GDI) must stretch or compress units in the logical coordinate system to fit units in the device coordinate system. For example, if the x-extent of the window is 2 and the x-extent of the viewport is 4, GDI converts two logical units (measured from the x-axis) into four device units. Similarly, if the y-extent of the window is 2 and the y-extent of the viewport is -1, GDI converts two logical units (measured from the y-axis) into one device unit.

The extents also define the relative orientation of the x- and y-axes in both coordinate systems. If the signs of matching window and viewport extents are the same, the axes have the same orientation. If the signs are different, the orientation is reversed. For example, if the y-extent of the window is 2 and the y-extent of the viewport is -1, GDI converts the positive y-axis in the logical coordinate system to the negative y-axis in the device coordinate system. If the x-extents are 2 and 4, GDI converts the positive x-axis in the logical coordinate system to the positive x-axis in the device coordinate system.

### Example

The following example uses the SetMapMode, SetWindowExt, and SetViewportExt functions to create a client area that is 10 logical units wide and 10 logical units high and then draws a rectangle that is 4 units wide and 4 units high:

```
HDC hdc;
RECT rc;

GetClientRect(hwnd, &rc);
hdc = GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 10, 10);
SetViewportExt(hdc, rc.right, rc.bottom);
Rectangle(hdc, 3, 3, 7, 7);
ReleaseDC(hwnd, hdc);
```

See Also

[GetWindowExt](#), [SetViewportExt](#), [SetWindowExtEx](#)

### SetWindowExtEx (3.1)

```
BOOL SetWindowExtEx(hdc, nX, nY, lpSize)
HDC hdc;          /* handle of device context */
int nX;          /* x-extent of window */
/
int nY;          /
/* y-extent of window
/
SIZE FAR* lpSize;
/* address of struct. for prev. extents
/
```

The SetWindowExtEx function sets the x- and y-extents of the window associated with the specified device context. The window, along with the viewport, defines how points are mapped from logical coordinates to device coordinates.

Parameter	Description
hdc	Identifies the device context.
nX	Specifies the x-extent, in logical units, of the window.
nY	Specifies the y-extent, in logical units, of the window.
lpSize	Points to a <u>SIZE</u> structure. The previous extents of the window (in logical units) are placed in this structure. If lpSize is NULL nothing is returned.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

When the following mapping modes are set, calls to the SetWindowExtEx and SetViewportExt functions are ignored:

MM\_HIENGLISH  
MM\_HIMETRIC  
MM\_LOENGLISH  
MM\_LOMETRIC  
MM\_TEXT  
MM\_TWIPS

When MM\_ISOTROPIC mode is set, an application must call the SetWindowExtEx function before calling SetViewportExt.

#### See Also

SetViewportExtEx, SetWindowExt

## SetWindowOrg (2.x)

```
DWORD SetWindowOrg(hdc, nXOrigin, nYOrigin)
HDC hdc;          /* handle of device context          */
int nXOrigin;     /* x-
coordinate to map to upper-left window corner
/
int nYOrigin;
* y-coordinate to map to upper-left window corner
/
```

The SetWindowOrg function sets the window origin for the given device context.

Parameter	Description
hdc	Identifies the device context.
nXOrigin	Specifies the logical x-coordinate to map to the upper-left corner of the window.
nYOrigin	Specifies the logical y-coordinate to map to the upper-left corner of the window.

### Returns

The return value is the coordinates of the previous window origin, in logical units, if the function is successful. The low-order word contains the x-coordinate of the previous window origin; the high-order word contains the y-coordinate. Otherwise, the return value is zero.

### Comments

The window origin is the origin of the logical coordinate system for a window. By changing the window origin, an application can change the way the graphics device interface (GDI) converts logical coordinates to device coordinates (the viewport). GDI converts logical coordinates to the device coordinates of the viewport in the same way as it converts the origin.

To convert points to the right, an application can specify a negative value for the nXOrigin parameter. Similarly, to convert points down (in the MM\_TEXT mapping mode), the nYOrigin parameter can be negative.

### Example

The following example uses the CopyMetaFile function to copy a metafile to a specified file, plays the copied metafile, uses the GetMetaFile function to retrieve a handle of the copied metafile, uses the SetWindowOrg function to change the position at which the metafile is played 200 logical units to the right, and then plays the metafile at the new location:

```
HANDLE hmf, hmfSource, hmfOld;
LPSTR lpzFile1 = "MFTest";

hmf = CopyMetaFile(hmfSource, lpzFile1);
PlayMetaFile(hdc, hmf);
DeleteMetaFile(hmf);

hmfOld = GetMetaFile(lpzFile1);
SetWindowOrg(hdc, -200, 0);
PlayMetaFile(hdc, hmfOld);

DeleteMetaFile(hmfSource);
DeleteMetaFile(hmfOld);
```

### See Also

CopyMetaFile, GetMetaFile, GetWindowOrg, PlayMetaFile, SetViewportOrg, SetWindowOrgEx

### SetWindowOrgEx (3.1)

```
BOOL SetWindowOrgEx(hdc, nX, nY, lpPoint)
HDC hdc;          /* handle of device context */
int nX;          /* x-coordinate of window */
int nY;          /* y-coordinate of window */
POINT FAR* lpPoint;
/* address of struct. for prev. origin */
```

The SetWindowOrgEx function sets the window origin of the specified device context. The window, along with the viewport, defines how points are mapped from logical coordinates to device coordinates.

Parameter	Description
hdc	Identifies the device context.
nX	Specifies the logical x-coordinate of the new origin of the window.
nY	Specifies the logical y-coordinate of the new origin of the window.
lpPoint	Points to a POINT structure. The previous origin of the window is placed in this structure. If lpPoint is NULL nothing is returned.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### See Also

[GetWindowOrg](#), [GetWindowOrgEx](#), [SetViewportOrgEx](#), [SetWindowOrg](#)



### SpoolFile (3.1)

HANDLE SpoolFile(lpszPrinter, lpszPort, lpszJob, lpszFile)

LPSTR lpszPrinter; /\* printer name \*/

LPSTR lpszPort; /\* port name \*/

LPSTR lpszJob;

\* job name

/

LPSTR lpszFile;

\* file name

/

The SpoolFile function puts a file into the spooler queue. This function is typically used by device drivers.

Parameter	Description
lpszPrinter	Points to a null-terminated string specifying the printer name--for example, "HP LasterJet IIP".
lpszPort	Points to a null-terminated string specifying the local name--for example, "LPT1:". This must be a local port.
lpszJob	Points to a null-terminated string specifying the name of the print job for the spooler. This string cannot be longer than 32 characters, including the null-terminating character.
lpszFile	Points to a null-terminated string specifying the path and filename of the file to put in the spooler queue. This file contains raw printer data.

#### Returns

The return value is the global handle that is passed to the spooler, if the function is successful. Otherwise, it is an error value, which can be one of the following:

SP\_APPABORT

SP\_ERROR

SP\_NOTREPORTED

SP\_OUTOFDISK

SP\_OUTOFMEMORY

SP\_USERABORT

#### Comments

Applications should ensure that the spooler is enabled before calling the SpoolFile function.

### StartDoc (3.1)

```
int StartDoc(hdc, lpd)
HDC hdc;          /* handle of device context */
DOCINFO FAR* lpd; /* pointer to DOCINFO structure */
/
```

The StartDoc function starts a print job. For Windows version 3.1, this function replaces the STARTDOC printer escape.

Parameter	Description
hdc	Identifies the device context for the print job.
lpdi	Points to a <u>DOCINFO</u> structure containing the name of the document file and the name of the output file.

#### Returns

The return value is positive if the function is successful. Otherwise, it is SP\_ERROR.

#### Comments

Applications should call the StartDoc function immediately before beginning a print job. Using this function ensures that documents containing more than one page are not interspersed with other print jobs.

The StartDoc function should not be used inside metafiles.

#### See Also

EndDoc, Escape, DOCINFO

## StartPage (3.1)

```
int StartPage(hdc)
HDC hdc;      /* handle of device context */
```

The StartPage function prepares the printer driver to accept data.

Parameter	Description
hdc	Identifies the device context for the print job.

### Returns

The return value is greater than zero if the function is successful. It is less than or equal to zero if an error occurs.

### Comments

The system disables the ResetDC function between calls to the StartPage and EndPage functions. This means that applications cannot change the device mode except at page boundaries.

### See Also

EndPage, Escape, ResetDC

## StretchBlt (2.x)

```
BOOL StretchBlt(hdcDest, nXOriginDest, nYOriginDest, nWidthDest, nHeightDest, hdcSrc,
               nXOriginSrc, nYOriginSrc, nWidthSrc, nHeightSrc, fdwRop)
HDC hdcDest;          /* destination device-context handle */
int nXOriginDest;    /* x-
coordinate of origin of destination rectangle
/
int nYOriginDest;    /* y-
coordinate of origin of destination rectangle
/
int nWidthDest;      /* width of destination rectangle
/
int nHeightDest;     /* height of destination rectangle
/
HDC hdcSrc;          /* source device-context handle
/
int nXOriginSrc;     /* x-
coordinate of origin of source rectangle
/
int nYOriginSrc;     /* y-
coordinate of origin of source rectangle
/
int nWidthSrc;       /* width of source rectangle
/
int nHeightSrc;     /* height of source rectangle
/
DWORD fdwRop;       /* raster operation
/
```

The StretchBlt function copies a bitmap from a source rectangle into a destination rectangle, stretching or compressing the bitmap if necessary to fit the dimensions of the destination rectangle. The StretchBlt function uses the stretching mode of the destination device context (set by the [SetStretchBltMode](#) function) to determine how to stretch or compress the bitmap.

Parameter	Description						
hdcDest	Identifies the device context to receive the bitmap.						
nXOriginDest	Specifies the logical x-coordinate of the upper-left corner of the destination rectangle.						
nYOriginDest	Specifies the logical y-coordinate of the upper-left corner of the destination rectangle.						
nWidthDest	Specifies the width, in logical units, of the destination rectangle.						
nHeightDest	Specifies the height, in logical units, of the destination rectangle.						
hdcSrc	Identifies the device context that contains the source bitmap.						
nXOriginSrc	Specifies the logical x-coordinate of the upper-left corner of the source rectangle.						
nYOriginSrc	Specifies the logical y-coordinate of the upper-left corner of the source rectangle.						
nWidthSrc	Specifies the width, in logical units, of the source rectangle.						
nHeightSrc	Specifies the height, in logical units, of the source rectangle.						
fdwRop	Specifies the raster operation to be performed. Raster-operation codes define how the graphics device interface ( <a href="#">GDI</a> ) combines colors in output operations that involve a current brush, a possible source bitmap, and a destination bitmap. This parameter can be one of the following values:						
	<table><thead><tr><th>Code</th><th>Description</th></tr></thead><tbody><tr><td>BLACKNESS</td><td>Turns all output black.</td></tr><tr><td>DSTINVERT</td><td>Inverts the destination bitmap.</td></tr></tbody></table>	Code	Description	BLACKNESS	Turns all output black.	DSTINVERT	Inverts the destination bitmap.
Code	Description						
BLACKNESS	Turns all output black.						
DSTINVERT	Inverts the destination bitmap.						

MERGECOPY	Combines the pattern and the source bitmap by using the Boolean AND operator.
MERGEPAINT	Combines the inverted source bitmap with the destination bitmap by using the Boolean OR operator.
NOTSRCCOPY	Copies the inverted source bitmap to the destination.
NOTSRCERASE	Inverts the result of combining the destination and source bitmaps by using the Boolean OR operator.
PATCOPY	Copies the pattern to the destination bitmap.
PATINVERT	Combines the destination bitmap with the pattern by using the Boolean XOR operator.
PATPAINT	Combines the inverted source bitmap with the pattern by using the Boolean OR operator. Combines the result of this operation with the destination bitmap by using the Boolean OR operator.
SRCAND	Combines pixels of the destination and source bitmaps by using the Boolean AND operator.
SRCCOPY	Copies the source bitmap to the destination bitmap.
SRCERASE	Inverts the destination bitmap and combines the result with the source bitmap by using the Boolean AND operator.
SRCINVERT	Combines pixels of the destination and source bitmaps by using the Boolean XOR operator.
SRCPAINT	Combines pixels of the destination and source bitmaps by using the Boolean OR operator.
WHITENESS	Turns all output white.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The StretchBlt function stretches or compresses the source bitmap in memory and then copies the result to the destination. If a pattern is to be merged with the result, it is not merged until the stretched source bitmap is copied to the destination.

If a brush is used, it is the selected brush in the destination device context.

The destination coordinates are transformed according to the destination device context; the source coordinates are transformed according to the source device context.

If the destination, source, and pattern bitmaps do not have the same color format, StretchBlt converts the source and pattern bitmaps to match the destination bitmaps. The foreground and background colors of the destination device context are used in the conversion.

If StretchBlt must convert a monochrome bitmap to color, it sets white bits (1) to the background color and black bits (0) to the foreground color. To convert color to monochrome, it sets pixels that match the background color to white (1) and sets all other pixels to black (0). The foreground and background colors of the device context with color are used.

StretchBlt creates a mirror image of a bitmap if the signs of the nWidthSrc and nWidthDest or nHeightSrc and nHeightDest parameters differ. If nWidthSrc and nWidthDest have different signs, the function creates a mirror image of the bitmap along the x-axis. If nHeightSrc and nHeightDest have different signs, the function creates a mirror image of the bitmap along the y-axis.

Not all devices support the StretchBlt function. Applications can discover whether a device supports StretchBlt by calling the [GetDeviceCaps](#) function and specifying the RASTERCAPS index.

#### Example

The following example retrieves the handle of the desktop window and uses it to create a device context. After retrieving the dimensions of the desktop window, the example calls the StretchBlt function to copy the desktop bitmap into a smaller rectangle in the destination device context.

```

HWND hwndDesktop;
HDC hdcLocal;
RECT rc;

hwndDesktop = GetDesktopWindow( );
hdcLocal = GetDC(hwndDesktop);

```

GetWindowRect(GetDesktopWindow(), &rc);

StretchBlt(hdc, 10, 10, 138, 106,  
hdcLocal, 0, 0, rc.right, rc.bottom, SRCCOPY);

ReleaseDC(hwndDesktop, hdcLocal);

See Also

BitBlt, GetDeviceCaps, SetStretchBltMode, StretchDIBits

### StretchDIBits (3.0)

```
int StretchDIBits(hdc, XDest, YDest, cxDest, cyDest, XSrc, YSrc, cxSrc, cySrc, lpvBits, lpbmi,
                 fuColorUse, fdwRop)
HDC hdc; /* handle of device context */
int XDest; /* x-
coordinate of destination rectangle
/
int YDest;
* y-coordinate of destination rectangle
/
int cxDest;
* width of destination rectangle
/
int cyDest;
* height of destination rectangle
/
int XSrc;
* x-coordinate of source rectangle
/
int YSrc;
* y-coordinate of source rectangle
/
int cxSrc;
* width of source rectangle
/
int cySrc;
* height of source rectangle
/
const void FAR* lpvBits;
* address of buffer with DIB bits
/
LPBITMAPINFO lpbmi;
* address of structure with bitmap data
/
UINT fuColorUse;
* RGB or palette indices
/
DWORD fdwRop;
* raster operation
/
```

The StretchDIBits function moves a device-independent bitmap (DIB) from a source rectangle into a destination rectangle, stretching or compressing the bitmap if necessary to fit the dimensions of the destination rectangle.

Parameter	Description
hdc	Identifies the destination device context for a screen surface or memory bitmap.
XDest	Specifies the logical x-coordinate of the destination rectangle.
YDest	Specifies the logical y-coordinate of the destination rectangle.
cxDest	Specifies the logical x-extent of the destination rectangle.
cyDest	Specifies the logical y-extent of the destination rectangle.
XSrc	Specifies the x-coordinate, in pixels, of the source rectangle in the DIB.
YSrc	Specifies the y-coordinate, in pixels, of the source rectangle in the DIB.
cxSrc	Specifies the width, in pixels, of the source rectangle in the DIB.
cySrc	Specifies the height, in pixels, of the source rectangle in the DIB.
lpvBits	Points to the DIB bits that are stored as an array of bytes.
lpbmi	Points to a <u>BITMAPINFO</u> structure that contains information about the DIB.
fuColorUse	Specifies whether the <u>bmiColors</u> member of the lpbmi parameter contains explicit <u>RGB</u> (red-green-blue) values or indices into the currently realized logical palette. The fuColorUse parameter can be one of the following values:

	Value	Meaning
	DIB_PAL_COLORS	The color table consists of an array of 16-bit indices into the currently realized logical palette.
	DIB_RGB_COLORS	The color table contains literal <u>RGB</u> values.
fdwRop		Specifies the raster operation to be performed. Raster-operation codes define how the graphics device interface ( <u>GDI</u> ) combines colors in output operations that involve a current brush, a possible <u>source</u> bitmap, and a destination bitmap. For a list of raster-operation codes, see the description of the <u>BitBlt</u> function.

#### Returns

The return value is the number of scan lines copied, if the function is successful.

#### Comments

The StretchDIBits function uses the stretching mode of the destination device context (set by the SetStretchBltMode function) to determine how to stretch or compress the bitmap.

The origin of the coordinate system for a device-independent bitmap is the lower-left corner. The origin of the coordinates of the destination rectangle depends on the current mapping mode of the device context.

StretchDIBits creates a mirror image of a bitmap if the signs of the cxSrc and cxDest parameters or the cySrc and cyDest parameters differ. If cxSrc and cxDest have different signs, the function creates a mirror image of the bitmap along the x-axis. If cySrc and cyDest have different signs, the function creates a mirror image of the bitmap along the y-axis.

#### See Also

SetMapMode, SetStretchBltMode, BITMAPINFO



## TextOut (2.x)

```
BOOL TextOut(hdc, nXStart, nYStart, lpszString, cbString)
HDC hdc; /* handle of device context */
int nXStart; /* x-coordinate
of starting position */
int nYStart; /* y-coordinate of starting position */
LPCSTR lpszString;
/* address of string */
int cbString;
/* number of bytes in string */
```

The TextOut function writes a character string at the specified location, using the currently selected font.

Parameter	Description
hdc	Identifies the device context.
nXStart	Specifies the logical x-coordinate of the starting point of the string.
nYStart	Specifies the logical y-coordinate of the starting point of the string.
lpszString	Points to the character string to be drawn.
cbString	Specifies the number of bytes in the string.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

Character origins are at the upper-left corner of the character cell.

By default, the TextOut function does not use or update the current position. If an application must update the current position when calling TextOut, it can call the SetTextAlign function with the wFlags parameter set to TA\_UPDATECP. When this flag is set, Windows ignores the nXStart and nYStart parameters on subsequent calls to the TextOut function, using the current position instead.

### Example

The following example uses the GetTextFace function to retrieve the face name of the current font, calls SetTextAlign so that the current position is updated when the TextOut function is called, and then writes some introductory text and the face name by calling TextOut:

```
int nFaceNameLen;
char aFaceName[80];

nFaceNameLen = GetTextFace(hdc, /* returns length of string */
    sizeof(aFaceName), /* size of face-name buffer */
    (LPSTR) aFaceName); /* address of face-name buffer */

SetTextAlign(hdc,
    TA_UPDATECP); /* updates current position */
MoveTo(hdc, 100, 100); /* sets current position*/
TextOut(hdc, 0, 0, /* uses current position for text */
    "This is the current face name: ", 31);
TextOut(hdc, 0, 0, aFaceName, nFaceNameLen);
```

### See Also

ExtTextOut, GetTextExtent, SetTextAlign, SetTextColor, TabbedTextOut

## UnrealizeObject (2.x)

```
BOOL UnrealizeObject(hgdiobj)
HGDIOBJ hgdiobj;      /* handle of brush or palette */
```

The `UnrealizeObject` function resets the origin of a brush or resets a logical palette. If the `hgdiobj` parameter identifies a brush, `UnrealizeObject` directs the system to reset the origin of the brush the next time it is selected. If the `hgdiobj` parameter identifies a logical palette, `UnrealizeObject` directs the system to realize the palette as though it had not previously been realized. The next time the application calls the `RealizePalette` function for the specified palette, the system completely remaps the logical palette to the system palette.

Parameter	Description
<code>hgdiobj</code>	Identifies the object to be reset.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The `UnrealizeObject` function should not be used with stock objects.

The `UnrealizeObject` function must be called whenever a new brush origin is set (by using the `SetBrushOrg` function).

A brush identified by the `hgdiobj` parameter must not be the currently selected brush of any device context.

A palette identified by `hgdiobj` can be the currently selected palette of a device context.

### Example

The following example uses the `SetBrushOrg` function to set the origin coordinates of the current brush to (3,5), uses the `SelectObject` function to remove that brush from the device context, uses the `UnrealizeObject` function to force the system to reset the origin of the specified brush, and then calls `SelectObject` again to select the brush into the device context with the new brush origin:

```
HBRUSH hbr, hbrOld;
SetBkMode(hdc, TRANSPARENT);
hbr = CreateHatchBrush(HS_CROSS, RGB(0, 0, 0));

UnrealizeObject(hbr);
SetBrushOrg(hdc, 0, 0);
hbrOld = SelectObject(hdc, hbr);

Rectangle(hdc, 0, 0, 200, 200);

hbr = SelectObject(hdc, hbrOld); /* deselects hbr */
UnrealizeObject(hbr); /* resets origin next time hbr selected */
SetBrushOrg(hdc, 3, 5);
hbrOld = SelectObject(hdc, hbr); /* selects hbr again */

Rectangle(hdc, 0, 0, 200, 200);

SelectObject(hdc, hbrOld);
DeleteObject(hbr);
```

### See Also

`RealizePalette`, `SelectObject`, `SetBrushOrg`

## UpdateColors (3.0)

```
int UpdateColors(hdc)
HDC hdc;      /* handle of device context */
```

The UpdateColors function updates the client area of the given device context by matching the current colors in the client area, pixel by pixel, to the system palette. An inactive window with a realized logical palette may call UpdateColors as an alternative to redrawing its client area when the system palette changes.

Parameter	Description
hdc	Identifies the device context.

**Returns**  
The return value is not used.

**Comments**  
Using UpdateColors to update a client area is typically faster than redrawing the area. However, because UpdateColors performs the color translation based on the color of each pixel before the system palette changed, each call to this function results in the loss of some color accuracy.

## GDI functions (3.1)

<u>AbortDoc</u>	Terminates a print job
<u>AddFontResource</u>	Adds a font to the font table
<u>AnimatePalette</u>	Replaces entries in a logical palette
<u>Arc</u>	Draws an arc
<u>BitBlt</u>	Copies a bitmap between device contexts
<u>Chord</u>	Draws a chord
<u>CloseMetaFile</u>	Closes a metafile DC and gets the handle
<u>CombineRgn</u>	Creates a region by combining two regions
<u>CopyMetaFile</u>	Copies a metafile
<u>CreateBitmap</u>	Creates a device-dependent memory bitmap
<u>CreateBitmapIndirect</u>	Creates a bitmap using <u>BITMAP</u> structure
<u>CreateBrushIndirect</u>	Creates a brush with the specified attributes
<u>CreateCompatibleBitmap</u>	Creates a bitmap compatible with the DC
<u>CreateCompatibleDC</u>	Creates a DC compatible with the specified DC
<u>CreateDC</u>	Creates a device context
<u>CreateDIBitmap</u>	Creates bitmap handle from DIB specification
<u>CreateDIBPatternBrush</u>	Creates a pattern brush from a DIB
<u>CreateDiscardableBitmap</u>	Creates discardable bitmap
<u>CreateEllipticRgn</u>	Creates an elliptical region
<u>CreateEllipticRgnIndirect</u>	Creates an elliptical region
<u>CreateFont</u>	Creates a logical font
<u>CreateFontIndirect</u>	Creates a font using a <u>LOGFONT</u> structure
<u>CreateHatchBrush</u>	Creates a hatched brush
<u>CreateIC</u>	Creates an information context
<u>CreateMetaFile</u>	Creates a metafile device context
<u>CreatePalette</u>	Creates a logical color palette
<u>CreatePatternBrush</u>	Creates a pattern brush from a bitmap
<u>CreatePen</u>	Creates a pen
<u>CreatePenIndirect</u>	Creates a pen using a <u>LOGPEN</u> structure
<u>CreatePolygonRgn</u>	Creates a polygonal region
<u>CreatePolyPolygonRgn</u>	Creates a region consisting of polygons
<u>CreateRectRgn</u>	Creates a rectangular region
<u>CreateRectRgnIndirect</u>	Creates a region using a <u>RECT</u> structure
<u>CreateRoundRectRgn</u>	Creates a rectangular region with round corners
<u>CreateScalableFontResource</u>	Creates a resource file with font info
<u>CreateSolidBrush</u>	Creates a solid brush with a specified color
<u>DeleteDC</u>	Deletes a device context
<u>DeleteMetaFile</u>	Invalidates a metafile handle
<u>DeleteObject</u>	Deletes an object from memory
<u>DeviceCapabilities</u>	Retrieves the capabilities of a device
<u>DeviceMode</u>	Displays a dialog box for printing modes
<u>DPtoLP</u>	Converts device points to logical points
<u>Ellipse</u>	Draws an ellipse
<u>EndDoc</u>	Ends a print job
<u>EndPage</u>	Ends a page
<u>EnumFontFamilies</u>	Retrieves fonts in a specified family
<u>EnumFonts</u>	Enumerates fonts on the specified device
<u>EnumMetaFile</u>	Enumerates metafile records
<u>EnumObjects</u>	Enumerates pens and brushes in a device context
<u>EqualRgn</u>	Compares two regions for equality
<u>Escape</u>	Allows access to capabilities device
<u>Printer escapes</u>	
<u>ExcludeClipRect</u>	Changes clipping region, excluding rectangle
<u>ExtDeviceMode</u>	Displays a dialog box for printing modes
<u>ExtFloodFill</u>	Fills an area with the current brush
<u>ExtTextOut</u>	Writes character string in rectangular region
<u>FillRgn</u>	Fills a region with the specified brush
<u>FloodFill</u>	Fills an area with the current brush
<u>FrameRgn</u>	Draws a border around a region
<u>GetAspectRatioFilter</u>	Retrieves setting of aspect-ratio filter
<u>GetAspectRatioFilterEx</u>	Retrieves setting of aspect-ratio filter
<u>GetBitmapBits</u>	Copies bitmap bits to a buffer
<u>GetBitmapDimension</u>	Retrieves the width and height of a bitmap

<u>GetBitmapDimensionEx</u>	Retrieves the width and height of a bitmap
<u>GetBkColor</u>	Retrieves the current background color
<u>GetBkMode</u>	Retrieves the background mode
<u>GetBoundsRect</u>	Returns current accumulated bounding rectangle
<u>GetBrushOrg</u>	Retrieves the origin of the current brush
<u>GetBrushOrgEx</u>	Retrieves the origin of the current brush
<u>GetCharABCWidths</u>	Retrieves the widths of TrueType characters
<u>GetCharWidth</u>	Retrieves the character widths
<u>GetClipBox</u>	Retrieves a rectangle for the clipping region
<u>GetCurrentPosition</u>	Retrieves the current position, in logical units
<u>GetCurrentPositionEx</u>	Retrieves the current position, in logical units
<u>GetDCOrg</u>	Retrieves translation origin for device context
<u>GetDeviceCaps</u>	Retrieves the device capabilities
<u>GetDIBits</u>	Copies the DIB bits into a buffer
<u>GetFontData</u>	Retrieves font metric data
<u>GetGlyphOutline</u>	Retrieves data for individual outline character
<u>GetKerningPairs</u>	Retrieves kerning pairs for the current font
<u>GetMapMode</u>	Retrieves the mapping mode
<u>GetMetaFile</u>	Creates a handle to a specified metafile
<u>GetMetaFileBits</u>	Creates a global memory object from a metafile
<u>GetNearestColor</u>	Retrieves the closest available color
<u>GetNearestPaletteIndex</u>	Retrieves the nearest match for a color
<u>GetObject</u>	Retrieves information about an object
<u>GetOutlineTextMetrics</u>	Retrieves metrics for TrueType fonts
<u>GetPaletteEntries</u>	Retrieves a range of palette entries
<u>GetPixel</u>	Retrieves RGB color value of specified pixel
<u>GetPolyFillMode</u>	Retrieves the current polygon-filling mode
<u>GetRasterizerCaps</u>	Retrieves status of TrueType fonts on system
<u>GetRgnBox</u>	Retrieves the bounding rectangle for a region
<u>GetROP2</u>	Retrieves the current drawing mode
<u>GetStockObject</u>	Retrieves handle of stock pen, brush, or font
<u>GetStretchBltMode</u>	Retrieves the current bitmap-stretching mode
<u>GetSystemPaletteEntries</u>	Retrieves entries from the system palette
<u>GetSystemPaletteUse</u>	Determines the use of an entire system palette
<u>GetTextCharacterExtra</u>	Retrieves the intercharacter spacing
<u>GetTextAlign</u>	Retrieves the text-alignment flags
<u>GetTextColor</u>	Retrieves the current text color
<u>GetTextExtent</u>	Determines dimensions of specified text string
<u>GetTextExtentPoint</u>	Retrieves dimensions of specified text string
<u>GetTextFace</u>	Retrieves the typeface name of the current font
<u>GetTextMetrics</u>	Retrieves the metrics for the current font
<u>GetViewportExt</u>	Retrieves the viewport extent
<u>GetViewportExtEx</u>	Retrieves the viewport extent
<u>GetViewportOrg</u>	Retrieves the viewport origin
<u>GetViewportOrgEx</u>	Retrieves the viewport origin
<u>GetWindowExt</u>	Retrieves the window extents
<u>GetWindowExtEx</u>	Retrieves the window extents
<u>GetWindowOrg</u>	Retrieves the window origin
<u>GetWindowOrgEx</u>	Retrieves the window origin
<u>IntersectClipRect</u>	Creates a clipping region from an intersection
<u>InvertRgn</u>	Inverts the colors in a region
<u>IsGDIObject</u>	Determines if a handle is not a GDI object
<u>LineDDA</u>	Computes successive points in a line
<u>LineTo</u>	Draws a line from the current position
<u>LPToDP</u>	Converts logical points to device points
<u>MoveTo</u>	Moves the current position
<u>MoveToEx</u>	Moves the current position
<u>OffsetClipRgn</u>	Moves a clipping region
<u>OffsetRgn</u>	Moves a region by a specified offset
<u>OffsetViewportOrg</u>	Moves the viewport origin
<u>OffsetViewportOrgEx</u>	Moves the viewport origin
<u>OffsetWindowOrg</u>	Moves the window origin
<u>OffsetWindowOrgEx</u>	Moves the window origin
<u>PaintRgn</u>	Fills region with brush in given device context

<u>PatBlt</u>	Creates a bitmap pattern
<u>Pie</u>	Draws a pie-shaped wedge
<u>PlayMetaFile</u>	Plays a metafile
<u>PlayMetaFileRecord</u>	Plays a metafile record
<u>Polygon</u>	Draws a polygon
<u>Polyline</u>	Draws line segments to connect specified points
<u>PolyPolygon</u>	Draws a series of polygons
<u>PtInRegion</u>	Determines whether a point is in a region
<u>PtVisible</u>	Determines whether point is in clipping region
<u>QueryAbort</u>	Determines whether to terminate a print job
<u>Rectangle</u>	Draws a rectangle
<u>RectInRegion</u>	Determines whether rectangle overlaps region
<u>RectVisible</u>	Determines whether rectangle is in clip region
<u>RemoveFontResource</u>	Removes an added font resource
<u>ResetDC</u>	Updates a device context
<u>ResizePalette</u>	Changes the size of a logical palette
<u>RestoreDC</u>	Restores the device context
<u>RoundRect</u>	Draws a rectangle with rounded corners
<u>SaveDC</u>	Saves the current state of a device context
<u>ScaleViewportExt</u>	Scales the viewport extents
<u>ScaleViewportExtEx</u>	Scales the viewport extents
<u>ScaleWindowExt</u>	Scales the window extents
<u>ScaleWindowExtEx</u>	Scales the window extents
<u>SelectClipRgn</u>	Selects clipping region for device context
<u>SelectObject</u>	Selects an object into a device context
<u>SetAbortProc</u>	Sets the abort function for a print job
<u>SetBitmapBits</u>	Sets the bitmap bits from an array of bytes
<u>SetBitmapDimension</u>	Sets the width and height of a bitmap
<u>SetBitmapDimensionEx</u>	Sets the width and height of a bitmap
<u>SetBkColor</u>	Sets the current background color
<u>SetBkMode</u>	Sets the background mode
<u>SetBoundsRect</u>	Controls the bounding-rectangle accumulation
<u>SetBrushOrg</u>	Sets the origin of the current brush
<u>SetDIBits</u>	Sets the bits of a bitmap
<u>SetDIBitsToDevice</u>	Sets DIB bits to a device
<u>SetMapMode</u>	Sets the mapping mode
<u>SetMapperFlags</u>	Sets the font-mapper flag
<u>SetMetaFileBits</u>	Creates a memory object from the metafile
<u>SetMetaFileBitsBetter</u>	Creates a memory object from the metafile
<u>SetPaletteEntries</u>	Sets the colors and flags for a color palette
<u>SetPixel</u>	Sets a pixel to the specified color
<u>SetPolyFillMode</u>	Sets the polygon-filling mode
<u>SetRectRgn</u>	Changes a region into a specified rectangle
<u>SetROP2</u>	Sets the current drawing mode
<u>SetStretchBltMode</u>	Sets the bitmap-stretching mode
<u>SetSystemPaletteUse</u>	Sets the use of system-palette static colors
<u>SetTextAlign</u>	Sets the text-alignment flags
<u>SetTextCharacterExtra</u>	Sets the intercharacter spacing
<u>SetTextColor</u>	Sets the foreground color for text
<u>SetTextJustification</u>	Sets the alignment for text output
<u>SetViewportExt</u>	Sets the viewport extents
<u>SetViewportExtEx</u>	Sets the viewport extents
<u>SetViewportOrg</u>	Sets the viewport origin
<u>SetViewportOrgEx</u>	Sets the viewport origin
<u>SetWindowExt</u>	Sets the window extents
<u>SetWindowExtEx</u>	Sets the window extents
<u>SetWindowOrg</u>	Sets the window origin
<u>SetWindowOrgEx</u>	Sets the window origin
<u>SpoolFile</u>	Puts a file in the spooler queue
<u>StartDoc</u>	Starts a print job
<u>StartPage</u>	Prepares a printer driver to receive data
<u>StretchBlt</u>	Copies a bitmap, transforming it if required
<u>StretchDIBits</u>	Moves DIB from source to destination rectangle
<u>TextOut</u>	Writes character string at specified location

UnrealizeObject  
UpdateColors

Resets brush origins and realizes palettes  
Updates colors in the client area

## DRV\_CLOSE (3.1)

### DRV\_CLOSE

The DRV\_CLOSE message is the first message sent by Windows to an installable driver after an application calls the CloseDriver function.

Parameter	Description
dwDriverIdentifier	Specifies the unique 32-bit identifier returned by the <u>OpenDriver</u> function.
hDriver	Identifies the instance of the installable driver that should be closed.
lParam1	Specifies driver-specific data.
lParam2	Specifies driver-specific data.

### Returns

An installable driver returns nonzero if its DriverProc function successfully closes the driver. Otherwise, it returns zero.

### Comments

The lParam1 and lParam2 parameters specify the same values as the lParam1 and lParam2 parameters for the CloseDriver function.

Each time a driver processes this message, it must decrement a private use-count variable. When the value of this variable is zero, Windows closes the driver.

### See Also

DRV\_OPEN



## DRV\_CONFIGURE (3.1)

### DRV\_CONFIGURE

The DRV\_CONFIGURE message is sent to inform an installable driver that it should display its private configuration dialog box.

Parameter	Description
dwDriverIdentifier	Specifies a unique 32-bit value that identifies the installable driver.
hDriver	Identifies an instance of the installable driver.
lParam1	Specifies the handle of the parent window for the configuration dialog box. This handle is in the parameter's low-order word.
lParam2	Points to an optional <u>DRVCONFIGINFO</u> structure. An installable driver should verify that this pointer is valid before using it.

#### Returns

An installable driver returns nonzero if it processes this message. Otherwise, it returns zero.

#### Comments

An installable driver that supports the DRV\_CONFIGURE message must provide its own dialog box template and dialog box procedure. It must also record the user's configuration requests in an appropriate file. (This may be the SYSTEM.INI file or some other file used by the driver for this purpose.)

#### See Also

[DRV\\_QUERYCONFIGURE](#)

## DRV\_DISABLE (3.1)

### DRV\_DISABLE

The DRV\_DISABLE message is the second message sent by Windows to an installable driver after an application calls the CloseDriver function.

Parameter	Description
dwDriverIdentifier	Not used.
hDriver	Identifies an instance of the installable driver.
lParam1	Not used.
lParam2	Not used.

#### Returns

An installable driver returns zero if it processes this message.

#### See Also

DRV\_CLOSE

## DRV\_ENABLE (3.1)

### DRV\_ENABLE

The DRV\_ENABLE message is sent to an installable driver when it is loaded or reloaded or whenever Windows is reinstalled after switching to an MS-DOS application.

Parameter	Description
dwDriverIdentifier	Not used.
hDriver	Identifies an instance of the installable driver.
lParam1	Not used.
lParam2	Not used.

#### Returns

An installable driver returns zero if it processes this message.

#### Comments

When the DriverProc function receives this message, it should initialize all of the driver-specific structures with default values.

#### See Also

DRV\_OPEN

## DRV\_EXITAPPLICATION (3.1)

### DRV\_EXITAPPLICATION

The DRV\_EXITAPPLICATION message is sent to all installable drivers when an application exits.

Parameter	Description						
dwDriverIdentifier	Specifies a unique 32-bit value that identifies the installable driver.						
IParam1	Specifies the type of application exit. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>DRVEA_NORMALEXIT</td><td>Set if the application terminated normally.</td></tr><tr><td>DRVEA_ABNORMALEXIT</td><td>Set if the application terminated abnormally (because of an application or system error).</td></tr></tbody></table>	Value	Meaning	DRVEA_NORMALEXIT	Set if the application terminated normally.	DRVEA_ABNORMALEXIT	Set if the application terminated abnormally (because of an application or system error).
Value	Meaning						
DRVEA_NORMALEXIT	Set if the application terminated normally.						
DRVEA_ABNORMALEXIT	Set if the application terminated abnormally (because of an application or system error).						
IParam2	Not used.						

#### Returns

The value returned by the application is ignored for this message.

#### See Also

[DRV\\_EXITSESSION](#)

## DRV\_EXITSESSION (3.1)

### DRV\_EXITSESSION

The DRV\_EXITSESSION message is sent to all installable drivers when Windows prepares to exit.

Parameter	Description
dwDriverIdentifier	Specifies a unique 32-bit value that identifies the installable driver.
IParam1	Reserved.
IParam2	Reserved.

#### Returns

The value returned by the application is ignored for this message.

#### Comments

The user interface and all other drivers are still enabled when this message is sent.

#### See Also

[DRV\\_EXITAPPLICATION](#)

## DRV\_FREE (3.1)

### DRV\_FREE

The DRV\_FREE message is the third message sent by Windows to an installable driver after an application calls the CloseDriver function.

Parameter	Description
dwDriverIdentifier	Not used.
hDriver	Identifies an instance of the installable driver.
lParam1	Not used.
lParam2	Not used.

#### Returns

An installable driver returns zero if it processes this message.

#### Comments

When an installable driver's DriverProc function receives this message, it should free the memory that was allocated for all driver-specific structures.

## DRV\_INSTALL (3.1)

### DRV\_INSTALL

The DRV\_INSTALL message is sent to an installable driver during the driver initialization process.

Parameter	Description
dwDriverIdentifier	Specifies a unique 32-bit value that identifies the installable driver.
hDriver	Identifies an instance of the installable driver.
lParam1	Not used.
lParam2	Points to an optional <u>DRVCONFIGINFO</u> structure. An installable driver should verify that this pointer is valid before using it.

#### Returns

An installable driver returns nonzero if it processes this message. Otherwise, it returns zero.

#### Comments

When the driver receives this message, it creates an entry for the driver in the SYSTEM.INI file and performs other necessary configuration operations.

## DRV\_LOAD (3.1)

### DRV\_LOAD

The DRV\_LOAD message is sent to an installable driver to notify the driver that it has been loaded.

Parameter	Description
dwDriverIdentifier	Not used.
hDriver	Identifies an instance of the installable driver.
lParam1	Not used.
lParam2	Not used.

#### Returns

An installable driver returns nonzero if its DriverProc function successfully loads the driver. Otherwise, it returns zero.



## DRV\_OPEN (3.1)

### DRV\_OPEN

The DRV\_OPEN message is sent to an installable driver each time it is opened.

Parameter	Description
dwDriverIdentifier	Specifies a unique 32-bit value that identifies the installable driver.
hDriver	Identifies an instance of the installable driver.
lParam1	Points to a null-terminated string containing any ASCII characters that followed the driver name in the SYSTEM.INI file.
lParam2	Contains the data specified by the lParam parameter, the third argument in the <u>OpenDriver</u> function.

#### Returns

An installable driver returns nonzero if it processes this message. Otherwise, it returns zero.

#### Comments

If no characters follow the driver name in SYSTEM.INI, the lParam1 parameter is a NULL pointer.

#### See Also

DRV\_CLOSE

## DRV\_QUERYCONFIGURE (3.1)

### DRV\_QUERYCONFIGURE

The DRV\_QUERYCONFIGURE message is sent to an installable driver to determine whether it can be configured by the user.

Parameter	Description
dwDriverIdentifier	Specifies a unique 32-bit value that identifies the installable driver.
hDriver	Identifies an instance of the installable driver.
lParam1	Not used.
lParam2	Not used.

#### Returns

An installable driver returns nonzero if it supports custom configuration and is capable of displaying a configuration dialog box. Otherwise, it returns zero.

#### See Also

[DRV\\_CONFIGURE](#)

## DRV\_POWER (3.1)

### DRV\_POWER

The DRV\_POWER message is sent to an installable driver each time the power supply to the associated device is about to be turned on or off.

Parameter	Description
dwDriverIdentifier	Specifies a unique 32-bit value that identifies the installable driver.
hDriver	Identifies an instance of the installable driver.
lParam1	Not used.
lParam2	Not used.

#### Returns

An installable driver returns nonzero if it processes this message. Otherwise, it returns zero.

## DRV\_REMOVE (3.1)

### DRV\_REMOVE

The DRV\_REMOVE message is sent by an application to an installable driver to notify the driver that it is about to be removed from the system.

Parameter	Description
dwDriverIdentifier	Specifies a unique 32-bit value that identifies the installable driver.
IParam1	Not used.
IParam2	Not used.

### Returns

An installable driver returns nonzero if it processes this message. Otherwise, it returns zero.

### Comments

When an installable driver receives this message, it should remove necessary entries from the SYSTEM.INI file.

## DRV\_USER (3.1)

### DRV\_USER

The DRV\_USER message is a user-defined or driver-dependent message.

Parameter	Description
dwDriverIdentifier	This parameter is not predefined; the value is driver dependent.
hDriver	This parameter is not predefined; the value is driver dependent.
lParam1	This parameter is not predefined; the value is driver dependent.
lParam2	This parameter is not predefined; the value is driver dependent.

### Returns

The return value is driver dependent.

### Installable-driver messages (3.1)

<u>DRV_CLOSE</u>	Indicates that driver should free resources
<u>DRV_CONFIGURE</u>	Indicates that driver should display dialog
<u>DRV_DISABLE</u>	Indicates that driver should unhook interrupts
<u>DRV_ENABLE</u>	Indicates that driver has been loaded or reloaded
<u>DRV_EXITAPPLICATION</u>	Indicates an application is exiting
<u>DRV_EXITSESSION</u>	Informs drivers that Windows is exiting
<u>DRV_FREE</u>	Indicates that driver must free all resources
<u>DRV_INSTALL</u>	Indicates that driver has been installed
<u>DRV_LOAD</u>	Indicates that driver has been loaded.
<u>DRV_OPEN</u>	Indicates that driver will be opened
<u>DRV_QUERYCONFIGURE</u>	Queries driver configuration capabilities
<u>DRV_POWER</u>	Indicates that device power-source was en/disabled
<u>DRV_REMOVE</u>	Indicates that driver will be removed
<u>DRV_USER</u>	Indicates that a user-defined action occurred

## \_hread (3.1)

```
long _hread(hf, hpvBuffer, cbBuffer)
HFILE hf;          /* file handle          */
void _huge* hpvBuffer;
address of buffer for read data
/
long cbBuffer;
* length of data buffer
/
```

The \_hread function reads data from the specified file. This function supports huge memory objects (that is, objects larger than 64K, allocated using the GlobalAlloc function).

Parameter	Description
hf	Identifies the file to be read.
hpvBuffer	Points to a buffer that is to receive the data read from the file.
cbBuffer	Specifies the number of bytes to be read from the file.

### Returns

The return value indicates the number of bytes that the function read from the file, if the function is successful. If the number of bytes read is less than the number specified in cbBuffer, the function reached the end of the file (EOF) before reading the specified number of bytes. The return value is -1L if the function fails.

### Comments

MS-DOS error return values are not available when an application calls this function.

### See Also

\_\_hread, hmemcpy, \_\_hwrite

## \_hwrite (3.1)

```
long _hwrite(hf, hpvBuffer, cbBuffer)
HFILE hf; /* file handle */
const void _huge* hpvBuffer; /* address of buffer for write data */
long cbBuffer; /* size of data */
```

The \_hwrite function writes data to the specified file. This function supports huge memory objects (that is, objects larger than 64K, allocated using the GlobalAlloc function).

Parameter	Description
hf	Identifies the file to be written to.
hpvBuffer	Points to a buffer that contains the data to be written to the file.
cbBuffer	Specifies the number of bytes to be written to the file.

### Returns

The return value indicates the number of bytes written to the file, if the function is successful. Otherwise, the return value is -1L.

### Comments

MS-DOS error return values are not available when an application calls this function.

### See Also

hmemcpy, hread, lwrite



## \_lclose (2.x)

```
HFILE _lclose(hf)
HFILE hf;      /* handle of file to close */
```

The \_lclose function closes the given file. As a result, the file is no longer available for reading or writing.

Parameter	Description
hf	Identifies the file to be closed. This handle is returned by the function that created or last opened the file.

### Returns

The return value is zero if the function is successful. Otherwise, it is `HFILE_ERROR`.

### Example

The following example copies a file to a temporary file, then closes both files:

```
int cbRead;
PBYTE pbBuf;

/* Allocate a buffer for file I/O. */
pbBuf = (PBYTE) LocalAlloc(LMEM_FIXED, 2048);

/* Copy the input file to the temporary file. */
do {
    cbRead = __lread(hfReadFile, pbBuf, 2048);
    __lwrite(hfTempFile, pbBuf, cbRead);
} while (cbRead != 0);

/* Free the buffer and close the files. */
LocalFree((HLOCAL) pbBuf);

_lclose(hfReadFile);
_lclose(hfTempFile);
```

### See Also

\_\_lopen, OpenFile

## \_lcreat (2.x)

```
HFILE _lcreat(lpszFilename, fnAttribute)
LPCSTR lpszFilename;          /* address of file to open */
int fnAttribute;              /* file attributes */
```

The \_lcreat function creates or opens a specified file. If the file does not exist, the function creates a new file and opens it for writing. If the file does exist, the function truncates the file size to zero and opens it for reading and writing. When the function opens the file, the pointer is set to the beginning of the file.

Parameter	Description										
lpszFilename	Points to a null-terminated string that names the file to be opened. The string must consist of characters from the Windows character set.										
fnAttribute	Specifies the file attributes. This parameter must be one of the following values:										
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Normal; can be read or written without restriction.</td></tr><tr><td>1</td><td>Read-only; cannot be opened for writing.</td></tr><tr><td>2</td><td>Hidden; not found by directory search.</td></tr><tr><td>3</td><td>System; not found by directory search.</td></tr></tbody></table>	Value	Meaning	0	Normal; can be read or written without restriction.	1	Read-only; cannot be opened for writing.	2	Hidden; not found by directory search.	3	System; not found by directory search.
Value	Meaning										
0	Normal; can be read or written without restriction.										
1	Read-only; cannot be opened for writing.										
2	Hidden; not found by directory search.										
3	System; not found by directory search.										

### Returns

The return value is a file handle if the function is successful. Otherwise, it is `HFILE_ERROR`.

### Comments

Use this function carefully. It is possible to open any file, even one that has already been opened by another function.

### Example

The following example uses the \_lcreat function to open a temporary file:

```
HFILE hfTempFile;
char szBuf[144];

/* Create a temporary file. */

GetTempFileName(0, "tst", 0, szBuf);

hfTempFile = _lcreat(szBuf, 0);

if (hfTempFile == HFILE_ERROR) {
    ErrorHandler();
}
```

## \_llseek (2.x)

```
LONG _llseek(hf, lOffset, nOrigin)
HFILE hf; /* file handle */
LONG lOffset; /* number of bytes to move */
/
int nOrigin; /* position to move from */
/
```

The \_llseek function repositions the pointer in a previously opened file.

Parameter	Description
hf	Identifies the file.
lOffset	Specifies the number of bytes the pointer is to be moved.
nOrigin	Specifies the starting position and direction of the pointer. This parameter must be one of the following values:
Value	Meaning
0	Move the file pointer lOffset bytes from the beginning of the file.
1	Move the file pointer lOffset bytes from its current position.
2	Move the file pointer lOffset bytes from the end of the file.

### Returns

The return value specifies the new offset, in bytes, of the pointer from the beginning of the file, if the function is successful. Otherwise, the return value is `HFILE_ERROR`.

### Comments

When a file is initially opened, the file pointer is positioned at the beginning of the file. The \_llseek function permits random access to a file's contents by moving the pointer an arbitrary amount without reading data.

### Example

The following example uses the \_llseek function to move the file pointer to the end of an existing file:

```
HFILE hfAppendFile;

/* Open the write file. */

hfAppendFile = _lopen("append.txt", WRITE);

/* Move to the end of the file. */

if (_llseek(hfAppendFile, 0L, 2) == -1) {
    ErrorHandler();
}
```

### See Also

\_lopen

## `_lopen (2.x)`

```
HFILE _lopen(lpszFilename, fnOpenMode)
LPCSTR lpszFilename;          /* address of file to open */
int fnOpenMode;              /* file access */
/
```

The `_lopen` function opens an existing file and sets the file pointer to the beginning of the file.

Parameter	Description																				
<code>lpszFilename</code>	Points to a null-terminated string that names the file to be opened. The string must consist of characters from the Windows character set.																				
<code>fnOpenMode</code>	Specifies the modes in which to open the file. This parameter consists of one access mode and an optional share mode. <table><thead><tr><th>Value</th><th>Access mode</th></tr></thead><tbody><tr><td><code>READ</code></td><td>Opens the file for reading only.</td></tr><tr><td><code>READ_WRITE</code></td><td>Opens the file for reading and writing.</td></tr><tr><td><code>WRITE</code></td><td>Opens the file for writing only.</td></tr></tbody></table> <table><thead><tr><th>Value</th><th>Share mode (optional)</th></tr></thead><tbody><tr><td><code>OF_SHARE_COMPAT</code></td><td>Opens the file in compatibility mode, allowing any process on a given machine to open the file any number of times. If the file has been opened by using any of the other sharing modes, <code>_lopen</code> fails.</td></tr><tr><td><code>OF_SHARE_DENY_NONE</code></td><td>Opens the file without denying other programs read or write access to the file. If the file has been opened in compatibility mode by any other program, <code>_lopen</code> fails.</td></tr><tr><td><code>OF_SHARE_DENY_READ</code></td><td>Opens the file and denies other programs read access to the file. If the file has been opened in compatibility mode or for read access by any other program, <code>_lopen</code> fails.</td></tr><tr><td><code>OF_SHARE_DENY_WRITE</code></td><td>Opens the file and denies other programs write access to the file. If the file has been opened in compatibility mode or for write access by any other program, <code>_lopen</code> fails.</td></tr><tr><td><code>OF_SHARE_EXCLUSIVE</code></td><td>Opens the file in exclusive mode, denying other programs both read and write access to the file. If the file has been opened in any other mode for read or write access, even by the current program, <code>_lopen</code> fails.</td></tr></tbody></table>	Value	Access mode	<code>READ</code>	Opens the file for reading only.	<code>READ_WRITE</code>	Opens the file for reading and writing.	<code>WRITE</code>	Opens the file for writing only.	Value	Share mode (optional)	<code>OF_SHARE_COMPAT</code>	Opens the file in compatibility mode, allowing any process on a given machine to open the file any number of times. If the file has been opened by using any of the other sharing modes, <code>_lopen</code> fails.	<code>OF_SHARE_DENY_NONE</code>	Opens the file without denying other programs read or write access to the file. If the file has been opened in compatibility mode by any other program, <code>_lopen</code> fails.	<code>OF_SHARE_DENY_READ</code>	Opens the file and denies other programs read access to the file. If the file has been opened in compatibility mode or for read access by any other program, <code>_lopen</code> fails.	<code>OF_SHARE_DENY_WRITE</code>	Opens the file and denies other programs write access to the file. If the file has been opened in compatibility mode or for write access by any other program, <code>_lopen</code> fails.	<code>OF_SHARE_EXCLUSIVE</code>	Opens the file in exclusive mode, denying other programs both read and write access to the file. If the file has been opened in any other mode for read or write access, even by the current program, <code>_lopen</code> fails.
Value	Access mode																				
<code>READ</code>	Opens the file for reading only.																				
<code>READ_WRITE</code>	Opens the file for reading and writing.																				
<code>WRITE</code>	Opens the file for writing only.																				
Value	Share mode (optional)																				
<code>OF_SHARE_COMPAT</code>	Opens the file in compatibility mode, allowing any process on a given machine to open the file any number of times. If the file has been opened by using any of the other sharing modes, <code>_lopen</code> fails.																				
<code>OF_SHARE_DENY_NONE</code>	Opens the file without denying other programs read or write access to the file. If the file has been opened in compatibility mode by any other program, <code>_lopen</code> fails.																				
<code>OF_SHARE_DENY_READ</code>	Opens the file and denies other programs read access to the file. If the file has been opened in compatibility mode or for read access by any other program, <code>_lopen</code> fails.																				
<code>OF_SHARE_DENY_WRITE</code>	Opens the file and denies other programs write access to the file. If the file has been opened in compatibility mode or for write access by any other program, <code>_lopen</code> fails.																				
<code>OF_SHARE_EXCLUSIVE</code>	Opens the file in exclusive mode, denying other programs both read and write access to the file. If the file has been opened in any other mode for read or write access, even by the current program, <code>_lopen</code> fails.																				

### Returns

The return value is a file handle if the function is successful. Otherwise, it is `HFILE_ERROR`.

### Example

The following example uses the `_lopen` function to open an input file:

```
HFILE hfReadFile;
/* Open the input file (read only). */

hfReadFile = _lopen("testfile", READ);

if (hfReadFile == HFILE_ERROR) {
    ErrorHandler();
}
```

### See Also

[OpenFile](#)

## \_lread (2.x)

```
UINT _lread(hf, hpvBuffer, cbBuffer)
HFILE hf;          /* file handle          */
void _huge* hpvBuffer; /* address of buffer for read data */
/
UINT cbBuffer;
* length of data buffer
/
```

The \_lread function reads data from the specified file.

Parameter	Description
hf	Identifies the file to be read.
hpvBuffer	Points to a buffer that is to receive the data read from the file.
cbBuffer	Specifies the number of bytes to be read from the file. This value cannot be greater than 0xFFFFE (65,534).

### Returns

The return value indicates the number of bytes that the function read from the file, if the function is successful. If the number of bytes read is less than the number specified in cbBuffer, the function reached the end of the file (EOF) before reading the specified number of bytes. The return value is HFILE\_ERROR if the function fails.

### Comments

MS-DOS error return values are not available when an application calls this function.

### Example

The following example uses the \_lread and \_lwrite functions to copy data from one file to another:

```
HFILE hfReadFile;
int cbRead;
PBYTE pbBuf;

/* Allocate a buffer for file I/O. */

pbBuf = (PBYTE) LocalAlloc(LMEM_FIXED, 2048);

/* Copy the input file to the temporary file. */

do {
    cbRead = _lread(hfReadFile, pbBuf, 2048);
    _lwrite(hfTempFile, pbBuf, cbRead);
} while (cbRead != 0);

/* Free the buffer and close the files. */

LocalFree((HLOCAL) pbBuf);

_lclose(hfReadFile);
_lclose(hfTempFile);

See Also
_hread, _lwrite
```

## \_lwrite (2.x)

```
UINT _lwrite(hf, hpvBuffer, cbBuffer)
HFILE hf; /* file handle */
const void _huge* hpvBuffer; /* address of buffer for write data */
/
UINT cbBuffer;
* size of data
/
```

The \_lwrite function writes data to the specified file.

Parameter	Description
hf	Identifies the file to be written to.
hpvBuffer	Points to a buffer that contains the data to be written to the file.
cbBuffer	Specifies the number of bytes to be written to the file. If this parameter is zero, the file is expanded or truncated to the current file-pointer position. This value cannot be greater than 0xFFFFE (65,534).

### Returns

The return value indicates the number of bytes written to the file, if the function is successful. Otherwise, the return value is HFILE\_ERROR.

### Comments

The buffer specified by hpvBuffer cannot extend past the end of a segment.

MS-DOS error return values are not available when an application calls this function.

### Example

The following example uses the \_lread and \_lwrite functions to copy data from one file to another:

```
int cbRead;
PBYTE pbBuf;

/* Allocate a buffer for file I/O. */
pbBuf = (PBYTE) LocalAlloc(LMEM_FIXED, 2048);

/* Copy the input file to the temporary file. */
do {
    cbRead = _lread(hfReadFile, pbBuf, 2048);
    _lwrite(hfTempFile, pbBuf, cbRead);
} while (cbRead != 0);

/* Free the buffer and close the files. */
LocalFree((HLOCAL) pbBuf);

_lclose(hfReadFile);
_lclose(hfTempFile);
```

### See Also

\_hwrite, \_lread

## AccessResource (2.x)

```
int AccessResource(hinst, hrsrc)
HINSTANCE hinst;          /* handle of module with resource */
HRSRC hrsrc;              /* handle of resource */
resource
/
```

The AccessResource function opens the given executable file and moves the file pointer to the beginning of the given resource.

Parameter	Description
hinst	Identifies the instance of the module whose executable file contains the resource.
hrsrc	Identifies the desired resource. This handle should be created by using the <u>FindResource</u> function.

### Returns

The return value is the handle of the resource file if the function is successful. Otherwise, it is -1.

### Comments

The AccessResource function supplies an MS-DOS file handle that can be used in subsequent file-read calls to load the resource. The file is opened for reading only.

Applications that use this function must close the resource file by calling the \_lclose function after reading the resource. AccessResource can exhaust available MS-DOS file handles and cause errors if the opened file is not closed after the resource is accessed.

In general, the LoadResource and LockResource functions are preferred. These functions will access the resource more quickly if several resources are being read, because Windows maintains a file-handle cache for accessing executable files. However, each call to AccessResource requires that a new handle be opened to the executable file.

You should not use AccessResource to access executable files that are installed in ROM on a ROM-based system, since there are no disk files associated with the executable file; in such a case, a file handle cannot be returned.

### See Also

FindResource, \_lclose, LoadResource, LockResource

## AddAtom (2.x)

```
ATOM AddAtom(lpszName)
LPCSTR lpszName; /* address of string to add */
```

The AddAtom function adds a character string to the local atom table and returns a unique value identifying the string.

Parameter	Description
lpszName	Points to the null-terminated character string to be added to the table.

### Returns

The return value specifies the newly created atom if the function is successful. Otherwise, it is zero.

### Comments

The AddAtom function stores no more than one copy of a given string in the atom table. If the string is already in the table, the function returns the existing atom value and increments (increases by one) the string's reference count.

The MAKEINTATOM macro can be used to convert a word value into a string that can be added to the atom table by using the AddAtom function.

The atom values returned by AddAtom are in the range 0xC000 through 0xFFFF.

Atoms are case-insensitive.

### Example

The following example uses the AddAtom function to add the string "This is an atom" to the local atom table:

```
ATOM at;
char szMsg[80];

at = AddAtom("This is an atom");

if (at == 0)
    MessageBox(hwnd, "AddAtom failed", "", MB_ICONSTOP);
else {
    wsprintf(szMsg, "AddAtom returned %u", at);
    MessageBox(hwnd, szMsg, "", MB_OK);
}
```

### See Also

DeleteAtom, FindAtom, GetAtomName, MAKEINTATOM



■ **AllocDStoCSAlias (3.0)**

```
UINT AllocDStoCSAlias(uSelector)
UINT uSelector;      /* data-segment selector */
```

The AllocDStoCSAlias function accepts a data-segment selector and returns a code-segment selector that can be used to execute code in the data segment.

Parameter	Description
uSelector	Specifies the data-segment selector.

**Returns**

The return value is the code-segment selector corresponding to the data-segment selector if the function is successful. Otherwise, it is zero.

**Comments**

The application must free the new selector by calling the FreeSelector function.

In protected mode, attempting to execute code directly in a data segment will cause a general-protection violation. AllocDStoCSAlias allows an application to execute code that the application had created in its own stack segment.

Windows does not track segment movements. Consequently, the data segment must be fixed and nondiscardable; otherwise, the data segment might move, invalidating the code-segment selector.

The PrestoChangoSelector function provides another method of obtaining a code selector corresponding to a data selector.

An application should not use this function unless it is absolutely necessary, since its use violates preferred Windows programming practices.

**See Also**

FreeSelector, PrestoChangoSelector

## Correction

The previous description of this function indicated that the application should free the selector with the FreeSelector function. Applications should not free the selector.

## AllocResource (2.x)

```
HGLOBAL AllocResource(hinst, hrsrc, cbResource)
HINSTANCE hinst;          /* handle of module containing resource */
HRSRC hrsrc;              /* handle of resource */
/
DWORD cbResource;
/* size to allocate, or zero */
/
```

The AllocResource function allocates uninitialized memory for the given resource.

Parameter	Description
hinst	Identifies the instance of the module whose executable file contains the resource.
hrsrc	Identifies the desired resource. This handle should have been created by using the <u>FindResource</u> function.
cbResource	Specifies the size, in bytes, of the memory object to allocate for the resource. If this parameter is zero, Windows allocates enough memory for the specified resource.

### Returns

The return value is the handle of the global memory object if the function is successful.

### See Also

FindResource, LoadResource

## AllocSelector (3.0)

```
UINT AllocSelector(uSelector)
UINT uSelector;          /* selector to copy or zero */
```

The AllocSelector function allocates a new selector.

Do not use this function in an application unless it is absolutely necessary, since its use violates preferred Windows programming practices.

Parameter	Description
uSelector	Specifies the selector to return. If this parameter specifies a valid selector, the function returns a new selector that is an exact copy of the one specified here. If this parameter is zero, the function returns a new, uninitialized sector.

### Returns

The return value is a selector that is either a copy of an existing selector, or a new, uninitialized selector. Otherwise, the return value is zero.

### Comments

The application must free the new selector by calling the FreeSelector function.

An application can call AllocSelector to allocate a selector that it can pass to the PrestoChangoSelector function.

### See Also

PrestoChangoSelector

## AnsiToOem (2.x)

```
void AnsiToOem(hpszWindows, hpszOem)
const char _huge* hpszWindows;          /* address of string to translate */
char _huge* hpszOem;                    /
* address of buffer for string
/
```

The AnsiToOem function translates a string from the Windows character set into the specified OEM character set.

Parameter	Description
hpszWindows	Points to a null-terminated string of characters from the Windows character set.
hpszOem	Points to the location where the translated string is to be copied. To translate the string in place, this parameter can be the same as hpszWindows.

### Returns

This function does not return a value.

### Comments

The string to be translated can be greater than 64K in length.

Windows-to-OEM mappings are defined by the keyboard driver, where this function is implemented. Some keyboard drivers may have different mappings than others, depending on the machine environment, and some keyboard driver support loading different OEM character sets; for example, the standard U.S. keyboard driver for an IBM keyboard supports loadable code pages, with the default being code page 437 and the most common alternative being code page 850. (The Windows character set is sometimes referred to as code page 1007.)

The OEM character set must always be used when accessing string data created by MS-DOS or MS-DOS applications. For example, a word processor should convert OEM characters to Windows characters when importing documents from an MS-DOS word processor. When an application makes an MS-DOS call, including a C run-time function call, filenames must be in the OEM character set, whereas they must be presented to the user in Windows characters (because the Windows fonts use Windows characters).

### Example

The following example is part of a dialog box in which a user would create a directory by typing a name in an edit control:

```
case IDOK:
    GetWindowText(GetDlgItem(hwndDlg, ID_EDITDIRNAME), szDirName,
        sizeof(szDirName));
    AnsiToOem(szDirName, szDirName);
    mkdir(szDirName);
    EndDialog(hwndDlg, 1);
    return TRUE;
```

### See Also

[AnsiToOemBuff](#), [OemToAnsi](#)

### AnsiToOemBuff (3.0)

```
void AnsiToOemBuff(lpszWindowsStr, lpszOemStr, cbWindowsStr)
LPCSTR lpszWindowsStr;      /* address of string to translate */
LPSTR lpszOemStr;           /* address of buffer for translated string */
UINT cbWindowsStr;         /* length of string to translate */
```

The `AnsiToOemBuff` function translates a string from the Windows character set into the specified OEM character set.

Parameter	Description
<code>lpszWindowsStr</code>	Points to a buffer containing one or more characters from the Windows character set.
<code>lpszOemStr</code>	Points to the location where the translated string is to be copied. To translate the string in place, this parameter can be the same as <code>lpszWindowsStr</code> .
<code>cbWindowsStr</code>	Specifies the number of bytes in the buffer identified by the <code>lpszWindowsStr</code> parameter. If <code>cbWindowsStr</code> is zero, the length is 64K (65,536).

#### Returns

This function does not return a value.

#### See Also

[AnsiToOem](#), [OemToAnsi](#)

## Catch (2.x)

```
int Catch(lpCatchBuf)
int FAR* lpCatchBuf;    /* address of buffer for array */
```

The Catch function captures the current execution environment and copies it to a buffer. The Throw function can use this buffer later to restore the execution environment. The execution environment includes the state of all system registers and the instruction counter.

Parameter	Description
lpCatchBuf	Points to a memory buffer large enough to contain a <u>CATCHBUF</u> array.

### Returns

The Catch function returns immediately with a return value of zero. When the Throw function is called, it returns again, this time with the return value specified in the nErrorReturn parameter of the Throw function.

### Comments

The Catch function is similar to the C run-time function setjmp.

### Example

The following example calls the Catch function to save the current execution environment before calling a recursive sort function. The first return value from Catch is zero. If the doSort function calls the Throw function, execution will again return to the Catch function. This time, Catch will return the STACKOVERFLOW error passed by the doSort function. The doSort function is recursive--that is, it calls itself. It maintains a variable, wStackCheck, that is used to check to see how much stack space has been used. If more than 3K of the stack has been used, doSort calls Throw to drop out of all the nested function calls back into the function that called Catch.

```
#define STACKOVERFLOW 1

UINT uStackCheck;
CATCHBUF catchbuf;

{
    int iReturn;
    char szBuf[80];

    if ((iReturn = Catch((int FAR*) catchbuf)) != 0) {
        . /* Error processing goes here. */
        .
        }
    else {
        uStackCheck = 0; /* initializes stack-usage count */
        doSort(1, 100); /* calls sorting function */
        }
    break;
}

void doSort(int sLeft, int sRight)
{
    int sLast;

    /*
    * Determine whether more than 3K of the stack has been
    * used, and if so, call Throw to drop back into the
    * original calling application.
    *
    * The stack is incremented by the size of the two parameters,
    * the two local variables, and the return value (2 for a near
    * function call).
    */

    uStackCheck += (sizeof(int) * 4) + 2;
```

```
    if (uStackCheck > (3 * 1024))
Throw((int FAR*) catchbuf, STACKOVERFLOW);
    .
    . /* A sorting algorithm goes here. */
    .

    doSort(sLeft, sLast - 1); /* note recursive call*/
    uStackCheck -= 10; /* updates stack-check variable */
}
```

See Also

Throw



## CloseSound (2.x)

void CloseSound(void)

This function is obsolete. Use the multimedia audio functions instead. For information about these functions, see the [Microsoft Windows Multimedia Programmer's Reference](#).

## CountVoiceNotes (2.x)

```
int CountVoiceNotes(nvoice)
int nvoice;      /* sound queue to be counted */
```

This function is obsolete. Use the multimedia audio functions instead. For information about these functions, see the [Microsoft Windows Multimedia Programmer's Reference](#).

## DebugBreak (3.0)

void DebugBreak(void)

The DebugBreak function causes a breakpoint exception to occur in the caller. This allows the calling process to signal the debugger, forcing it to take some action. If the process is not being debugged, the system invokes the default breakpoint exception handler. This may cause the calling process to terminate.

### Returns

This function does not return a value.

### Comments

This function is the only way to break into a WEP (Windows exit procedure) in a dynamic-link library.

For more information about using the debugging functions with Microsoft debugging tools, see Tools

### Example

The following example uses the DebugBreak function to signal the debugger immediately before the application handles the WM\_DESTROY message:

```
case WM_DESTROY:  
    DebugBreak();  
    PostQuitMessage(0);  
    break;
```

See Also

WEP

## DebugOutput (3.1)

```
void FAR _cdecl DebugOutput(flags, lpszFmt, ...)
UINT flags; /* type of message */
LPCSTR lpszFmt; /* address of
formatting string
/
```

The DebugOutput function sends a message to the debugging terminal. Applications can apply the formatting codes to the message string and use filters and options to control the message category.

Parameter	Description										
flags	Specifies the type of message to be sent to the debugging terminal. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><b><u>DBF_TRACE</u></b></td><td>The message reports that no error has occurred and supplies information that may be useful during debugging. Example: "t Kernel: LoadResource(14AE of GDI)"</td></tr><tr><td><b><u>DBF_WARNING</u></b></td><td>The message reports a situation that may or may not be an error, depending on the circumstances. Example: "wn Kernel: GlobalWire(17BE of GDI) (try GlobalLock)"</td></tr><tr><td><b><u>DBF_ERROR</u></b></td><td>The message reports an error resulting from a failed call to a Windows function. The application continues to run. Example: "err Kernel: LocalShrink(15EA of GDI) (invalid local heap)"</td></tr><tr><td><b><u>DBF_FATAL</u></b></td><td>The message reports an error that will terminate the application. Example: "fat! User: SetDeskWallpaper(16CA of USER)"</td></tr></tbody></table>	Value	Meaning	<b><u>DBF_TRACE</u></b>	The message reports that no error has occurred and supplies information that may be useful during debugging. Example: "t Kernel: LoadResource(14AE of GDI)"	<b><u>DBF_WARNING</u></b>	The message reports a situation that may or may not be an error, depending on the circumstances. Example: "wn Kernel: GlobalWire(17BE of GDI) (try GlobalLock)"	<b><u>DBF_ERROR</u></b>	The message reports an error resulting from a failed call to a Windows function. The application continues to run. Example: "err Kernel: LocalShrink(15EA of GDI) (invalid local heap)"	<b><u>DBF_FATAL</u></b>	The message reports an error that will terminate the application. Example: "fat! User: SetDeskWallpaper(16CA of USER)"
Value	Meaning										
<b><u>DBF_TRACE</u></b>	The message reports that no error has occurred and supplies information that may be useful during debugging. Example: "t Kernel: LoadResource(14AE of GDI)"										
<b><u>DBF_WARNING</u></b>	The message reports a situation that may or may not be an error, depending on the circumstances. Example: "wn Kernel: GlobalWire(17BE of GDI) (try GlobalLock)"										
<b><u>DBF_ERROR</u></b>	The message reports an error resulting from a failed call to a Windows function. The application continues to run. Example: "err Kernel: LocalShrink(15EA of GDI) (invalid local heap)"										
<b><u>DBF_FATAL</u></b>	The message reports an error that will terminate the application. Example: "fat! User: SetDeskWallpaper(16CA of USER)"										
lpszFmt	Points to a formatting string identical to the formatting strings used by the Windows function wsprintf. This string must be less than 160 characters long. Any additional formatting can be done by supplying additional parameters following lpszFmt.										
...	Specifies zero or more optional arguments. The number and type of arguments depends on the corresponding format-control character sequences specified in the lpszFmt parameter.										

### Returns

This function does not return a value.

### Comments

The messages sent by the DebugOutput function are affected by the system debugging options and trace-filter flags that are set and retrieved by using the GetWinDebugInfo and SetWinDebugInfo functions. These options and flags are stored in a WINDEBUGINFO structure.

Unlike most other Windows functions, DebugOutput uses the C calling convention (\_cdecl), rather than the Pascal calling convention. As a result, the caller must pop arguments off the stack. Also, arguments must be pushed on the stack from right to left. In C-language modules, the C compiler performs this task.

### See Also

GetWinDebugInfo, OutputDebugString, SetWinDebugInfo, wsprintf, WINDEBUGINFO

DBF\_TRACE 0x0000

The message reports that no error has occurred and supplies information that may be useful during debugging. Example: "t Kernel: LoadResource(14AE of GDI)"

DBF\_TRACE 0x0000

DBF\_WARNING 0x4000

The message reports a situation that may or may not be an error, depending on the circumstances.  
Example: "wn Kernel: GlobalWire(17BE of GDI) (try GlobalLock)"

DBF\_WARNING 0x4000



DBF\_ERROR 0x8000

The message reports an error resulting from a failed call to a Windows function. The application continues to run. Example: "err Kernel: LocalShrink(15EA of GDI) (invalid local heap)"

DBF\_ERROR 0x8000

DBF\_FATAL 0xc000

The message reports an error that will terminate the application. Example: "fatl User: SetDeskWallpaper (16CA of USER)"

DBF\_FATAL 0xc000

## DeleteAtom (2.x)

```
ATOM DeleteAtom(atm)
ATOM atm;      /* atom to delete */
```

The DeleteAtom function decrements (decreases by one) the reference count of a local atom by one. If the atom's reference count is reduced to zero, the string associated with the atom is removed from the local atom table.

An atom's reference count specifies the number of times the atom has been added to the atom table. The AddAtom function increments (increases by one) the count on each call. DeleteAtom decrements the count on each call and removes the string only if the atom's reference count is reduced to zero.

Parameter	Description
atm	Identifies the atom and character string to be deleted.

### Returns

The return value is zero if the function is successful. Otherwise, it is equal to the atm parameter.

### Comments

The only way to ensure that an atom has been deleted from the atom table is to call this function repeatedly until it fails. When the count is decremented to zero, the next call to the FindAtom or DeleteAtom function will fail.

DeleteAtom has no effect on integer atoms (atoms created by using the MAKEINTATOM macro). The function always returns zero for integer atoms.

### Example

The following example uses the DeleteAtom function to decrement the reference count for the specified atom:

```
ATOM at;

at = DeleteAtom(atTest);

if (at == NULL)
    MessageBox(hwnd, "atom count decremented",
    "DeleteAtom", MB_OK);
else
    MessageBox(hwnd, "atom count could not be decremented",
    "DeleteAtom", MB_ICONEXCLAMATION);
```

### See Also

AddAtom, FindAtom, GlobalDeleteAtom

## DirectedYield (3.1)

```
void DirectedYield(htask)  
HTASK htask;
```

The DirectedYield function puts the current task to sleep and awakens the given task.

Parameter	Description
htask	Specifies the task to be executed.

### Returns

This function does not return a value.

### Comments

When relinquishing control to other applications (that is, when exiting hard mode), a Windows-based debugger should call DirectedYield, identifying the handle of the task being debugged. This ensures that the debugged application runs next and that messages received during debugging are processed by the appropriate windows.

The Windows scheduler executes a task only when there is an event waiting for it, such as a paint message, or a message posted in the message queue.

If an application uses DirectedYield for a task with no events scheduled, the task will not be executed. Instead, Windows searches the task queue. In some cases, however, you may want the application to force a specific task to be scheduled. The application can do this by calling the PostAppMessage function, specifying a WM\_NULL message identifier. Then, when the application calls DirectedYield, the scheduler will run the task regardless of the task's event status.

DirectedYield starts the task identified by htask at the location where it left off. Typically, debuggers should use TaskSwitch instead of DirectedYield, because TaskSwitch can start a task at any address.

DirectedYield returns when the current task is reawakened. This occurs when the task identified by htask waits for messages or uses the Yield or DirectedYield function. Execution will continue as before the task switch.

DirectedYield is located in KRNL286.EXE and KRNL386.EXE and is available in Windows versions 3.0 and 3.1.

### See Also

PostAppMessage, TaskSwitch, TaskGetCSIP, TaskSetCSIP, Yield

## DOS3Call (3.0)

### DOS3Call

The DOS3Call function allows an application to call an MS-DOS Interrupt 21h function. DOS3Call can be called only from assembly-language routines. It is exported from KRNL286.EXE and KRNL386.EXE and is not defined in any Windows header or include files.

#### Parameters

Registers must be set up as required by the desired Interrupt 21h function before the application calls the DOS3Call function.

#### Returns

The register contents are preserved as they are returned by the Interrupt 21h function.

#### Comments

Applications should use this function instead of a directly coded MS-DOS Interrupt 21h function. The DOS3Call function runs somewhat faster than the equivalent MS-DOS Interrupt 21h function running in Windows.

#### Example

The following example shows how to prototype the DOS3Call function in C:

```
extern void FAR PASCAL DOS3Call(void);
```

To declare the DOS3Call function in an assembly-language routine, an application could use the following line:

```
extrn DOS3CALL: far
```

If the application includes CMACROS.INC, the function is declared as follows:

```
extrnFP DOS3Call
```

The following example is a typical use of the DOS3Call function:

```
extrn DOS3CALL: far
.
.
.
    ; set registers

    movah, DOSFUNC    ;DOSFUNC = Int 21h function number
    cCall    DOS3Call
```

## FatalAppExit (3.0)

```
void FatalAppExit(fuAction, lpszMessageText)
UINT fuAction;          /* must be zero          */
LPCSTR lpszMessageText;
/* string to display in message box
/
```

The FatalAppExit function displays a message box and terminates the application when the message box is closed. If the user is running the debugging version of the Windows operating system, the message box gives the user the opportunity to terminate the application or to cancel the message box and return to the caller.

Parameter	Description
fuAction	Reserved; must be zero.
lpszMessageText	Points to a null-terminated string that is displayed in the message box. The message is displayed on a single line. To accommodate low-resolution screens, the string should contain no more than 35 characters.

### Returns

This function does not return a value.

### Comments

An application should call the FatalAppExit function only when it is incapable of terminating any other way. FatalAppExit may not always free an application's memory or close its files, and it may cause a general failure of Windows. An application that encounters an unexpected error should terminate by freeing all its memory and returning from its main message loop.

### See Also

[FatalExit](#), [TerminateApp](#)



## FatalExit (2.x)

```
void FatalExit(nErrCode)
int nErrCode;      /* error value to display */
```

The FatalExit function sends the current state of Windows to the debugger and prompts for instructions on how to proceed.

An application should call this function for debugging purposes only; it should not call the function in a retail version of the application. Calling this function in the retail version will terminate the application.

Parameter	Description
nErrCode	Specifies the error value to be displayed.

### Returns

This function does not return a value.

### Comments

The displayed information includes an error value followed by a symbolic stack trace, showing the flow of execution up to the point of the call.

The FatalExit function prompts the user to respond to an Abort, Break, or Ignore message. Windows processes the response as follows:

Response	Description
A (Abort)	Terminate immediately.
B (Break)	Enter the debugger.
I (Ignore)	Return to the caller.

You can specify any combination of error values for the nErrCode parameter, since the meaning of the values is unique to your application. However, the error value -1 must always be reserved for the stack-overflow message. When this value is specified, Windows automatically displays a stack-overflow message.

### See Also

[FatalAppExit](#)

## FindAtom (2.x)

```
ATOM FindAtom(lpszString)
LPCSTR lpszString;      /* address of string to find */
```

The FindAtom function searches the local atom table for the specified character string and retrieves the atom associated with that string.

Parameter	Description
lpszString	Points to the null-terminated character string to search for.

### Returns

The return value identifies the atom associated with the given string if the function is successful. Otherwise (if the string is not in the table), the return value is zero.

### Example

The following example uses the FindAtom function to retrieve the atom for the string "This is an atom":

```
ATOM at;
char szMsg[80];

if ((at = FindAtom("This is an atom")) == 0)
    MessageBox(hwnd, "could not find atom",
    "FindAtom", MB_ICONEXCLAMATION);
else {
    wsprintf(szMsg, "atom = %u", at);
    MessageBox(hwnd, szMsg, "FindAtom", MB_OK);
}
```

### See Also

AddAtom, DeleteAtom

## FindResource (2.x)

```
HRSRC FindResource(hinst, lpszName, lpszType)
HINSTANCE hinst;          /* handle of module containing resource */
LPCSTR lpszName;
/* address of resource name
/
LPCSTR lpszType;
/* address of resource type
/
```

The FindResource function determines the location of a resource in the specified resource file.

Parameter	Description
hinst	Identifies the instance of the module whose executable file contains the resource.
lpszName	Specifies the name of the resource. For details, see the following Comments section.
lpszType	Specifies the resource type. For details, see the following Comments section. For predefined resource types, this parameter should be one of the following values:

Value	Meaning
<u>RT_ACCELERATOR</u>	Accelerator table
<u>RT_BITMAP</u>	Bitmap resource
<u>RT_CURSOR</u>	Cursor resource
<u>RT_DIALOG</u>	Dialog box
<u>RT_FONT</u>	Font resource
<u>RT_FONTDIR</u>	Font directory resource
<u>RT_ICON</u>	Icon resource
<u>RT_MENU</u>	Menu resource
<u>RT_RCDATA</u>	User-defined resource (raw data)
<u>RT_STRING</u>	String resource

### Returns

The return value is the handle of the named resource if the function is successful. Otherwise, it is NULL.

### Comments

If the high-order word of the lpszName or lpszType parameter is zero, the low-order word specifies the integer identifier of the name or type of the given resource. Otherwise, the parameters are long pointers to null-terminated strings. If the first character of the string is a pound sign (#), the remaining characters represent a decimal number that specifies the integer identifier of the resource's name or type. For example, the string #258 represents the integer ID 258.

To reduce the amount of memory required for the resources used by an application, the application should refer to the resources by integer identifier instead of by name.

An application must not call the FindResource and LoadResource functions to load cursor, icon, and string resources. Instead, it must load these resources by calling the LoadCursor, LoadIcon, and LoadString functions, respectively.

Although the application can call the FindResource and LoadResource functions to load other predefined resource types, it should load the corresponding resources by calling the LoadAccelerators, LoadBitmap, and LoadMenu functions.

### See Also

LoadAccelerators, LoadBitmap, LoadCursor, LoadIcon, LoadMenu, LoadResource, LoadString

RT\_ACCELERATOR MAKEINTRESOURCE(9)

Accelerator table

RT\_ACCELERATOR MAKEINTRESOURCE(9)

RT\_BITMAP MAKEINTRESOURCE(2)

Bitmap resource

RT\_BITMAP MAKEINTRESOURCE(2)

RT\_CURSOR MAKEINTRESOURCE(1)

Cursor resource



RT\_CURSOR MAKEINTRESOURCE(1)

RT\_DIALOG MAKEINTRESOURCE(5)  
Dialog box

RT\_DIALOG MAKEINTRESOURCE(5)

RT\_FONT MAKEINTRESOURCE(8)

Font resource

RT\_FONT MAKEINTRESOURCE(8)

RT\_FONTDIR MAKEINTRESOURCE(7)

Font directory resource

RT\_FONTDIR MAKEINTRESOURCE(7)

RT\_ICON MAKEINTRESOURCE(3)

Icon resource



RT\_ICON MAKEINTRESOURCE(3)

RT\_MENU MAKEINTRESOURCE(4)

Menu resource

RT\_MENU MAKEINTRESOURCE(4)

RT\_RCDATA MAKEINTRESOURCE(10)

User-defined resource (raw data)

RT\_RC DATA MAKEINTRESOURCE(10)

RT\_STRING MAKEINTRESOURCE(6)  
String resource

RT\_STRING MAKEINTRESOURCE(6)

## FreeLibrary (2.x)

```
void FreeLibrary(hinst)
HINSTANCE hinst;          /* handle of loaded library module */
```

The FreeLibrary function decrements (decreases by one) the reference count of the loaded library module. When the reference count reaches zero, the memory occupied by the module is freed.

Parameter	Description
hinst	Identifies the loaded library module.

### Returns

This function does not return a value.

### Comments

A dynamic-link library (DLL) must not call the FreeLibrary function within its WEP function (Windows exit procedure).

The reference count for a library module is incremented (increased by one) each time an application calls the LoadLibrary function for the library module.

### Example

The following example uses the LoadLibrary function to load TOOLHELP.DLL and the FreeLibrary function to free it:

```
HINSTANCE hinstToolHelp = LoadLibrary("TOOLHELP.DLL");

if ((UINT) hinstToolHelp > 32) {
    . /* use GetProcAddress to use TOOLHELP functions */
    .
}
else {
    ErrorHandler();
}

if ((UINT) hinstToolHelp > 32)
    FreeLibrary(hinstToolHelp); /* free TOOLHELP.DLL */
```

### See Also

GetProcAddress, LoadLibrary, WEP



## FreeModule (3.0)

```
BOOL FreeModule(hinst)  
HINSTANCE hinst;      /* handle of loaded module */
```

The FreeModule function decrements (decreases by one) the reference count of the loaded module. When the reference count reaches zero, the memory occupied by the module is freed.

Parameter	Description
hinst	Identifies the loaded module.

### Returns

The return value is zero if the reference count is decremented to zero and the module's memory is freed. Otherwise, the return value is nonzero.

### Comments

The reference count for a module is incremented (increased by one) each time an application calls the LoadModule function for the module.

### See Also

LoadModule

## FreeProcInstance (2.x)

```
void FreeProcInstance(lpProc)
FARPROC lpProc;          /* instance address of function to free */
```

The FreeProcInstance function frees the specified function from the data segment bound to it by the MakeProcInstance function.

Parameter	Description
lpProc	Points to the procedure-instance address of the function to be freed. It must be created by using the <u>MakeProcInstance</u> function.

### Returns

This function does not return a value.

### Comments

After a procedure instance has been freed, attempts to call the function using the freed procedure-instance address will result in an unrecoverable error.

### See Also

MakeProcInstance

## FreeResource (2.x)

```
BOOL FreeResource(hglbResource)
HGLOBAL hglbResource;      /* handle of loaded resource */
```

The FreeResource function decrements (decreases by one) the reference count of a loaded resource. When the reference count reaches zero, the memory occupied by the resource is freed.

Parameter	Description
hglbResource	Identifies the data associated with the resource. The handle is assumed to have been created by using the <u>LoadResource</u> function.

### Returns

The return value is zero if the function is successful. Otherwise, it is nonzero, indicating that the function has failed and the resource has not been freed.

### Comments

The reference count for a resource is incremented (increased by one) each time an application calls the LoadResource function for the resource.

### See Also

LoadResource

## FreeSelector (3.0)

```
UINT FreeSelector(uSelector)
UINT uSelector;          /* selector to be freed */
```

The FreeSelector function frees a selector originally allocated by the AllocSelector or AllocDStoCSAlias function. After the application calls this function, the selector is invalid and must not be used.

An application should not use this function unless it is absolutely necessary, since its use violates preferred Windows programming practices.

Parameter	Description
uSelector	Specifies the selector to be freed.

### Returns

The return value is zero if the function is successful. Otherwise, it is the selector specified by the uSelector parameter.

### Comments

The limit for the selector specified by the uSelector parameter must not be larger than 64K. If the limit of the selector exceeds 64K, the FreeSelector function may free selectors that are still required by the program.

### See Also

AllocDStoCSAlias, AllocSelector

## GetAtomHandle (2.x)

```
HLOCAL GetAtomHandle(atm)
ATOM atm; /* atom to retrieve handle of */
```

The GetAtomHandle function retrieves a handle of the specified atom.

This function is only provided for compatibility with Windows, versions 1.x and 2.x. It should not be used with Windows 3.0 and later.

Parameter	Description
atm	Specifies an atom whose handle is to be retrieved.

### Returns

The return value is a handle of the specified atom if the function is successful.

### See Also

[GetAtomName](#), [GlobalGetAtomName](#)

## GetAtomName (2.x)

```
UINT GetAtomName(atm, lpszBuffer, cbBuffer)
ATOM atm;          /* atom identifying character string */
LPSTR lpszBuffer; /* address of
buffer for atom string
/
int cbBuffer;
* size of buffer
/
```

The GetAtomName function retrieves a copy of the character string associated with the specified local atom.

Parameter	Description
atom	Specifies the local atom that identifies the character string to be retrieved.
lpszBuffer	Points to the buffer for the character string.
cbBuffer	Specifies the maximum size, in bytes, of the buffer.

### Returns

The return value specifies the number of bytes copied to the buffer, if the function is successful.

### Comments

The string returned for an integer atom (an atom created by the `MAKEINTATOM` macro) will be a null-terminated string, where the first character is a pound sign (#) and the remaining characters make up the `UINT` used in `MAKEINTATOM`.

### Example

The following example uses the GetAtomName function to retrieve the character string associated with a local atom:

```
char szBuf[80];

GetAtomName(atTest, szBuf, sizeof(szBuf));

MessageBox(hwnd, szBuf, "GetAtomName", MB_OK);
```

### See Also

[AddAtom](#), [DeleteAtom](#), [FindAtom](#), [MAKEINTATOM](#)

## GetCodeHandle (2.x)

```
HGLOBAL GetCodeHandle(lpProc)
FARPROC lpProc;          /* instance address of function */
```

The GetCodeHandle function determines which code segment contains the specified function.

Parameter	Description
lpProc	Points to the procedure-instance address of the function for which to return the code segment. Typically, this address is returned by the <u>MakeProcInstance</u> function.

### Returns

The return value identifies the code segment that contains the function if the GetCodeHandle function is successful. Otherwise, it is NULL.

### Comments

If the code segment that contains the function is already loaded, the GetCodeHandle function marks the segment as recently used. If the code segment is not loaded, GetCodeHandle attempts to load it. Thus, an application can use this function to attempt to preload one or more segments necessary to perform a particular task.

### See Also

MakeProcInstance

### GetCodeInfo (3.0)

```
void GetCodeInfo(lpProc, lpSegInfo)
FARPROC lpProc;          /* function address or module handle */
SEGINFO FAR* lpSegInfo; /* address of structure for segment information */
/
```

The GetCodeInfo function retrieves a pointer to a structure containing information about a code segment.

Parameter	Description
lpProc	Specifies the procedure-instance address of the function (typically, returned by the <u>MakeProcInstance</u> function) in the segment for which information is to be retrieved, or it specifies a module handle (typically, returned by the <u>GetModuleHandle</u> function) and segment number.
lpSegInfo	Points to a <u>SEGINFO</u> structure that will be filled with information about the code segment.

#### Returns

This function does not return a value.

#### See Also

GetModuleHandle, MakeProcInstance



## GetCurrentPDB (3.0)

UINT GetCurrentPDB(void)

The GetCurrentPDB function returns the selector address of the current MS-DOS program database (PDB), also known as the program segment prefix (PSP).

### Returns

The return value is the selector address of the current PDB if the function is successful.

### Example

The following example uses the GetCurrentPDB function to list the current command tail:

```
typedef struct {
    WORD pspInt20; /* Int 20h instruction */
    WORD pspNextParagraph; /* segment addr. of next paragraph */
    BYTE res1; /* reserved */
    BYTE pspDispatcher[5]; /* long call to MS-DOS */
    DWORD pspTerminateVector; /* termination address (Int 22h) */
    DWORD pspControlCVector; /* addr of CTRL+C (Int 23h) */
    DWORD pspCritErrorVector; /* addr of Crit-Error (Int 24h) */
    WORD res2[11]; /* reserved */
    WORD pspEnvironment; /* segment address of environment */
    WORD res3[23]; /* reserved */
    BYTE pspFCB_1[16]; /* default FCB #1 */
    BYTE pspFCB_2[16]; /* default FCB #2 */
    DWORD res4; /* reserved */
    BYTE pspCommandTail[128]; /* command tail (also default DTA) */
} PSP, FAR* LPSP;

LPSP lpsp = (LPSP) MAKELP(GetCurrentPDB(), 0);

MessageBox(NULL, lpsp->pspCommandTail, "PDB Command Tail", MB_OK);
```

## GetCurrentTask (2.x)

HTASK GetCurrentTask(void)

The GetCurrentTask function retrieves the handle of the current (running) task.

### Returns

The return value is a handle of the current task if the function is successful. Otherwise, it is NULL.

## GetDOSEnvironment (3.0)

LPSTR GetDOSEnvironment(void)

The GetDOSEnvironment function returns a far pointer to the environment string of the current (running) task.

### Returns

The return value is a far pointer to the current environment string.

### Comments

Unlike an application, a dynamic-link library (DLL) does not have a copy of the environment string. As a result, the library must call this function to retrieve the environment string.

### Example

The following example uses the GetDOSEnvironment function to return a pointer to the environment, and then lists the environment settings:

```
LPSTR lpszEnv;
```

```
lpszEnv = GetDOSEnvironment();
while (*lpszEnv != '\0') {
    . /* process the environment string */
    .
    /* Move to the next environment string */

    lpszEnv += lstrlen(lpszEnv) + 1;
}
```

## GetDriveType (3.0)

```
UINT GetDriveType(DriveNumber)
int DriveNumber; /* 0 = A, 1 = B, and so on */
```

The GetDriveType function determines whether a disk drive is removable, fixed, or remote.

Parameter	Description
DriveNumber	Specifies the drive for which the type is to be determined (0 = drive A, 1 = drive B, 2 = drive C, and so on).

### Returns

The return value is DRIVE\_REMOVABLE (disk can be removed from the drive), DRIVE\_FIXED (disk cannot be removed from the drive), or DRIVE\_REMOTE (drive is a remote, or network, drive), if the function is successful. Otherwise, the return value is zero.

### Example

The following example uses the GetDriveType function to determine the drive type for all possible disk drives (letters A through Z):

```
int iDrive;
WORD wReturn;
char szMsg[80];

for (iDrive = 0, wReturn = 0;
     (iDrive < 26) && (wReturn != 1); iDrive++) {

    wReturn = GetDriveType(iDrive);

    sprintf(szMsg, "drive %c: ", iDrive + 'A');

    switch (wReturn) {
    case 0:
        strcat(szMsg, "undetermined");
        break;

    case DRIVE_REMOVABLE:
        strcat(szMsg, "removable");
        break;

    case DRIVE_FIXED:
        strcat(szMsg, "fixed");
        break;

    case DRIVE_REMOTE:
        strcat(szMsg, "remote (network)");
        break;
    }
    TextOut(hdc, 10, 15 * iDrive, szMsg, strlen(szMsg));
}
}
```

## GetFreeSpace (3.0)

```
DWORD GetFreeSpace(fuFlags)  
UINT fuFlags; /* ignored in Windows 3.1 */
```

The GetFreeSpace function scans the global heap and returns the number of bytes of memory currently available.

Parameter	Description
fuFlags	This parameter is ignored in Windows 3.1.

### Returns

The return value is the amount of available memory, in bytes.

### Comments

The amount of memory specified by the return value is not necessarily contiguous; the GlobalCompact function returns the number of bytes in the largest block of free global memory.

In standard mode, the value returned represents the number of bytes in the global heap that are not used and that are not reserved for code.

In 386-enhanced mode, the return value is an estimate of the amount of memory available to an application. It does not account for memory held in reserve for non-Windows applications.

### See Also

GlobalCompact

## GetInstanceData (2.x)

```
int GetInstanceData(hinst, npbData, cbData)
HINSTANCE hinst;          /* handle of previous instance */
BYTE* npbData;           /* address of current instance data buffer */
/*
int cbData;
* number of bytes to transfer
*/
```

The GetInstanceData function copies data from a previous instance of an application into the data area of the current instance.

Parameter	Description
hinst	Identifies a previous instance of the application.
npbData	Points to a buffer in the current instance.
cbData	Specifies the number of bytes to be copied.

### Returns

The return value specifies the number of bytes copied if the function is successful. Otherwise, it is zero.

## GetKBCodePage (3.0)

int GetKBCodePage(void)

The GetKBCodePage function returns the current Windows code page.

### Returns

The return value specifies the code page currently loaded by Windows, if the function is successful. It can be one of the following values:

Value	Meaning
437	Default (United States, used by most countries: indicates that there is no OEMANSI.BIN in the Windows directory)
850	International (OEMANSI.BIN = XLAT850.BIN)
860	Portugal (OEMANSI.BIN = XLAT860.BIN)
861	Iceland (OEMANSI.BIN = XLAT861.BIN)
863	French Canadian (OEMANSI.BIN = XLAT863.BIN)
865	Norway/Denmark (OEMANSI.BIN = XLAT865.BIN)

### Comments

The keyboard driver provides the GetKBCodePage function. An application using this function must include the following information in its module-definition (.DEF) file:

```
IMPORTS
    KEYBOARD.GETKBCODEPAGE
```

If the OEMANSI.BIN file is in the Windows directory, Windows reads it and overwrites the OEM/ANSI translation tables in the keyboard driver.

When the user selects a language from the Setup program and the language does not use the default code page (437), Setup copies the appropriate file (such as XLAT850.BIN) to OEMANSI.BIN in the Windows system directory. If the language uses the default code page, Setup deletes OEMANSI.BIN, if it exists, from the Windows system directory.

### Example

The following example uses the GetKBCodePage function to display the current code page:

```
char szBuf[80];
int i, cp, subtype, f_keys, len;

char *apszKeyboards[] = {
    "IBM PX/XT",
    "Olivetti ICO",
    "IBM AT",
    "IBM Enhanced",
    "Nokia 1050",
    "Nokia 9140",
    "Standard Japanese",
};

cp = GetKBCodePage();

if ((i = GetKeyboardType(0)) == 0 || i > 7) {
    MessageBox(NULL, "invalid keyboard type",
        "GetKeyboardType", MB_ICONSTOP);
    break;
}

subtype = GetKeyboardType(1);
f_keys = GetKeyboardType(2);

len = wsprintf(szBuf, "%s keyboard, subtype %d\n",
    apszKeyboards[i - 1], subtype);
len = wsprintf(szBuf + len, "%d function keys, code page %d",
    f_keys, cp);
```

MessageBox(NULL, szBuf, "Keyboard Information", MB\_OK);

See Also

GetKeyboardType



## GetKeyboardType (3.0)

```
int GetKeyboardType(fnKeybInfo)
int fnKeybInfo; /* specifies type of information to retrieve */
```

The GetKeyboardType function retrieves information about the current keyboard.

Parameter	Description								
fnKeybInfo	Determines the type of keyboard information to be retrieved. This parameter can be one of the following values:								
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Retrieves the keyboard type.</td></tr><tr><td>1</td><td>Retrieves the keyboard subtype.</td></tr><tr><td>2</td><td>Retrieves the number of function keys on the keyboard.</td></tr></tbody></table>	Value	Meaning	0	Retrieves the keyboard type.	1	Retrieves the keyboard subtype.	2	Retrieves the number of function keys on the keyboard.
Value	Meaning								
0	Retrieves the keyboard type.								
1	Retrieves the keyboard subtype.								
2	Retrieves the number of function keys on the keyboard.								

### Returns

The return value specifies the requested information if the function is successful. Otherwise, it is zero.

### Comments

The subtype is an OEM-dependent value. The subtype may be one of the following values:

Value	Meaning
1	IBM PC/XT, or compatible (83-key) keyboard
2	Olivetti "ICO" (102-key) keyboard
3	IBM AT (84-key) or similar keyboard
4	IBM Enhanced (101- or 102-key) keyboard
5	Nokia 1050 and similar keyboards
6	Nokia 9140 and similar keyboards
7	Japanese keyboard

The keyboard driver provides the GetKeyboardType function. An application using this function must include the following information in its module-definition (.DEF) file:

```
IMPORTS
    KEYBOARD.GETKEYBOARDTYPE
```

The application can also determine the number of function keys on a keyboard from the keyboard type. The number of function keys for each keyboard type follows:

Type	Number of function keys
1	10
2	12 (sometimes 18)
3	10
4	12
5	10
6	24
7	This value is hardware-dependent and must be specified by the OEM.

### Example

The following example uses the GetKeyboardType function to display information about the current keyboard:

```
char szBuf[80];
int i, cp, subtype, f_keys, len;

char *apszKeyboards[] = {
    "IBM PX/XT",
    "Olivetti ICO",
    "IBM AT",
    "IBM Enhanced",
    "Nokia 1050",
    "Nokia 9140",
```

```
    "Standard Japanese",  
};  
  
cp = GetKBCodePage();  
  
if ((i = GetKeyboardType(0)) == 0 || i > 7) {  
    MessageBox(NULL, "invalid keyboard type",  
    "GetKeyboardType", MB_ICONSTOP);  
    break;  
}  
  
subtype = GetKeyboardType(1);  
f_keys = GetKeyboardType(2);  
  
len = wsprintf(szBuf, "%s keyboard, subtype %d\n",  
    apszKeyboards[i - 1], subtype);  
len = wsprintf(szBuf + len, " %d function keys, code page %d",  
    f_keys, cp);  
  
MessageBox(NULL, szBuf, "Keyboard Information", MB_OK);
```

### GetKeyNameText (3.0)

```
int GetKeyNameText(IParam, lpszBuffer, cbMaxKey)
LONG IParam;          /* 32-bit parameter of keyboard message */
LPSTR lpszBuffer;     /*
address of a buffer for key name
/
int cbMaxKey;
* specifies maximum key string length
/
```

The GetKeyNameText function retrieves a string that represents the name of a key.

Parameter	Description								
IParam	Specifies the 32-bit parameter of the keyboard message (such as <u>WM_KEYDOWN</u> ) to be processed. The GetKeyNameText function interprets the following portions of IParam: <table><thead><tr><th>Bits</th><th>Meaning</th></tr></thead><tbody><tr><td>16-23</td><td>Character scan code.</td></tr><tr><td>24</td><td>Extended bit. Distinguishes some keys on an enhanced keyboard.</td></tr><tr><td>25</td><td>"Don't care" bit. The application calling this function sets this bit to indicate that the function should not distinguish between left and right CTRL and SHIFT keys, for example.</td></tr></tbody></table>	Bits	Meaning	16-23	Character scan code.	24	Extended bit. Distinguishes some keys on an enhanced keyboard.	25	"Don't care" bit. The application calling this function sets this bit to indicate that the function should not distinguish between left and right CTRL and SHIFT keys, for example.
Bits	Meaning								
16-23	Character scan code.								
24	Extended bit. Distinguishes some keys on an enhanced keyboard.								
25	"Don't care" bit. The application calling this function sets this bit to indicate that the function should not distinguish between left and right CTRL and SHIFT keys, for example.								
lpszBuffer	Points to a buffer that will receive the key name.								
cbMaxKey	Specifies the maximum length, in bytes, of the key name, not including the terminating null character (this parameter should be one less than the size of the buffer pointed to by the lpszBuffer parameter).								

#### Returns

The return value is the length, in bytes, of the string copied to the specified buffer, if the function is successful. Otherwise, it is zero.

#### Comments

The format of the key-name string depends on the current keyboard driver. This driver maintains a list of names in the form of character strings for keys with names longer than a single character. The key name is translated, according to the layout of the currently installed keyboard, into the principal language supported by the keyboard driver.

#### Correction

The previous documentation incorrectly listed bit 21 of lparam as the extended bit and bit 22 as the "don't care" bit.

## GetModuleFileName (2.x)

```
int GetModuleFileName(hinst, lpzFilename, cbFileName)
HINSTANCE hinst;          /* handle of module          */
LPSTR lpzFilename;      /* address of buffer for filename */
/* address of buffer for filename
/
int cbFileName;
* maximum number of bytes to copy
/
```

The GetModuleFileName function retrieves the full path and filename of the executable file from which the specified module was loaded.

Parameter	Description
hinst	Identifies the module or the instance of the module.
lpzFilename	Points to the buffer that is to receive the null-terminated filename.
cbFileName	Specifies the maximum number of bytes to copy, including the terminating null character. The filename is truncated if it is longer than cbFileName. This parameter should be set to the length of the filename buffer.

### Returns

The return value specifies the length, in bytes, of the string copied to the specified buffer, if the function is successful. Otherwise, it is zero.

### Example

The following example retrieves an application's filename by using the instance handle passed to the application in the WinMain function:

```
int PASCAL WinMain(HINSTANCE hinst, HINSTANCE hPrevInst,
    LPSTR lpCmdLine, int nCmdShow)
{
    char szModuleName[260];

    GetModuleFileName(hinst, szModuleName, sizeof(szModuleName));
}
```

### See Also

GetModuleHandle

## GetModuleHandle (2.x)

```
HMODULE GetModuleHandle(lpszModuleName)  
LPCSTR lpszModuleName;          /* address of name of module */
```

The GetModuleHandle function retrieves the handle of the specified module.

Parameter	Description
lpszModuleName	Points to a null-terminated string that specifies the name of the module.

### Returns

The return value is the handle of the module if the function is successful. Otherwise, it is NULL.

### See Also

[GetModuleFileName](#)

## GetModuleUsage (2.x)

```
int GetModuleUsage(hinst)
HINSTANCE hinst;          /* handle of module */
```

The GetModuleUsage function retrieves the reference count of a specified module.

Parameter	Description
hinst	Identifies the module or an instance of the module.

### Returns

The return value specifies the reference count of the module if the function is successful.

### Comments

Windows increments (increases by one) a module's reference count each time an application calls the LoadModule function. The count is decremented (decreased by one) when an application calls the FreeModule function.

### See Also

FreeModule, LoadModule

## GetNumTasks (2.x)

UINT GetNumTasks(void)

The GetNumTasks function retrieves the number of currently running tasks.

### Returns

The return value specifies the number of current tasks.



### GetPrivateProfileInt (3.0)

```
UINT GetPrivateProfileInt(lpszSection, lpszEntry, default, lpszFilename)
LPCSTR lpszSection;          /* address of section          */
LPCSTR lpszEntry;           /* address of entry          */
/*
/
int default;
* return value if entry not found
/
LPCSTR lpszFilename;
* address of initialization filename
/
```

The GetPrivateProfileInt function retrieves the value of an integer from an entry within a specified section of a specified initialization file.

Parameter	Description
lpszSection	Points to a null-terminated string containing the section heading in the initialization file.
lpszEntry	Points to the null-terminated string containing the entry whose value is to be retrieved.
default	Specifies the default value to return if the entry cannot be found in the initialization file. This value must be a positive integer in the range 0 through 32,767 (0x0000 through 0x7FFF).
lpszFilename	Points to a null-terminated string that names the initialization file. If this parameter does not contain a full path, Windows searches for the file in the Windows directory.

#### Returns

The return value is the integer value of the specified entry if the function is successful. It is the value of the default parameter if the function does not find the entry. The return value is zero if the value that corresponds to the specified entry is not an integer.

#### Comments

The function searches the file for an entry that matches the name specified by the lpszEntry parameter under the section heading specified by the lpszSection parameter. An integer entry in the initialization file must have the following form:

```
[section]
entry=value
.
.
.
```

If the value that corresponds to the entry consists of digits followed by nonnumeric characters, the function returns the value of the digits. For example, the function would return 102 for the line "Entry=102abc".

The GetPrivateProfileInt function is not case-dependent, so the strings in the lpszSection and lpszEntry parameters may contain a combination of uppercase and lowercase letters.

GetPrivateProfileInt supports hexadecimal notation. When GetPrivateProfileInt is used to retrieve a negative integer, the value should be cast to an int.

An application can use the [GetProfileInt](#) function to retrieve an integer value from the WIN.INI file.

#### Example

The following example uses the GetPrivateProfileInt function to retrieve the last line number by reading the LastLine entry from the [MyApp] section of TESTCODE.INI:

```
WORD wInt;
char szMsg[144];

wInt = GetPrivateProfileInt("MyApp", "LastLine",
    0, "testcode.ini");

sprintf(szMsg, "last line was %d", wInt);
MessageBox(hwnd, szMsg, "GetPrivateProfileInt", MB_OK);
```

See Also

[GetPrivateProfileString](#), [GetProfileInt](#)

### Windows 3.1 changes

The `GetPrivateProfileInt` function supports hexadecimal notation. When the `GetPrivateProfileInt` function is used to retrieve a negative integer, the value should be cast to an `int`.

### GetPrivateProfileString (3.0)

```
int GetPrivateProfileString(lpszSection, lpszEntry, lpszDefault, lpszReturnBuffer, cbReturnBuffer,
                           lpszFilename)
LPCSTR lpszSection;      /* address of section */
LPCSTR lpszEntry;       /* address of entry */
LPCSTR lpszDefault;     /* address of default string */
LPSTR lpszReturnBuffer; /* address of destination buffer */
int cbReturnBuffer;     /* size of destination buffer */
LPCSTR lpszFilename;    /* address of initialization filename */
```

The GetPrivateProfileString function retrieves a character string from the specified section in the specified initialization file.

Parameter	Description
lpszSection	Points to a null-terminated string that specifies the section containing the entry.
lpszEntry	Points to the null-terminated string containing the entry whose associated string is to be retrieved. If this value is NULL, all entries in the section specified by the lpszSection parameter are copied to the buffer specified by the lpszReturnBuffer parameter. For more information, see the following Comments section.
lpszDefault	Points to a null-terminated string that specifies the default value for the given entry if the entry cannot be found in the initialization file. This parameter must never be NULL.
lpszReturnBuffer	Points to the buffer that receives the character string.
cbReturnBuffer	Specifies the size, in bytes, of the buffer pointed to by the lpszReturnBuffer parameter.
lpszFilename	Points to a null-terminated string that names the initialization file. If this parameter does not contain a full path, Windows searches for the file in the Windows directory.

#### Returns

The return value specifies the number of bytes copied to the specified buffer, not including the terminating null character.

#### Comments

The function searches the file for an entry that matches the name specified by the lpszEntry parameter under the section heading specified by the lpszSection parameter. If the entry is found, its corresponding string is copied to the buffer. If the entry does not exist, the default character string specified by the lpszDefault parameter is copied. A string entry in the initialization file must have the following form:

```
[section]
entry=string
```

```
.
.
.
```

If lpszEntry is NULL, the GetPrivateProfileString function copies all entries in the specified section to the supplied buffer. Each string will be null-terminated, with the final string ending with two zero-termination characters. If the supplied destination buffer is too small to hold all the strings, the last string will be truncated and followed with two zero-termination characters.

If the string associated with lpszEntry is enclosed in single or double quotation marks, the marks are discarded when GetPrivateProfileString returns the string.

GetPrivateProfileString is not case-dependent, so the strings in lpszSection and lpszEntry may contain a combination of uppercase and lowercase letters.

An application can use the GetProfileString function to retrieve a string from the WIN.INI file.

The lpszDefault parameter must point to a valid string, even if the string is empty (its first character is zero).

#### Example

The following example uses the GetPrivateProfileString function to determine the last file saved by the [MyApp] application by reading the LastFile entry in TESTCODE.INI:

```
char szMsg[144], szBuf[80];

GetPrivateProfileString("MyApp", "LastFile",
    "", szBuf, sizeof(szBuf), "testcode.ini");

sprintf(szMsg, "last file was %s", szBuf);
MessageBox(hwnd, szMsg, "GetPrivateProfileString", MB_OK);
```

See Also

GetProfileString, WritePrivateProfileString

## GetProcAddress (2.x)

```
FARPROC GetProcAddress(hinst, lpszProcName)
HINSTANCE hinst;          /* handle of module */
LPCSTR lpszProcName;     /* address of
function */
/
```

The GetProcAddress function retrieves the address of the given module function.

Parameter	Description
hinst	Identifies the module that contains the function.
lpszProcName	Points to a null-terminated string containing the function name, or specifies the ordinal value of the function. If it is an ordinal value, the value must be in the low-order word and the high-order word must be zero.

### Returns

The return value is the address of the module function's entry point if the GetProcAddress function is successful. Otherwise, it is NULL.

If the lpszProcName parameter is an ordinal value and a function with the specified ordinal does not exist in the module, GetProcAddress can still return a non-NULL value. In cases where the function may not exist, specify the function by name rather than ordinal value.

### Comments

Use the GetProcAddress function to retrieve addresses of exported functions in dynamic-link libraries (DLLs). The MakeProcInstance function can be used to access functions within different instances of the current module.

The spelling of the function name (pointed to by the lpszProcName parameter) must be identical to the spelling as it appears in the EXPORTS section of the source DLL's module-definition (.DEF) file.

### Example

The following example uses the GetProcAddress function to retrieve the address of the TimerCount function in TOOLHELP.DLL:

```
char szBuf[80];
TIMERINFO timerinfo;
HINSTANCE hinstToolHelp;
BOOL (FAR *lpfnTimerCount) (TIMERINFO FAR*);

/* Turn off the "File not found" error box. */

SetErrorMode(SEM_NOOPENFILEERRORBOX);

/* Load the TOOLHELP.DLL library module. */

hinstToolHelp = LoadLibrary("TOOLHELP.DLL");

if (hinstToolHelp > HINSTANCE_ERROR) { /* loaded successfully */

    /* Retrieve the address of the TimerCount function. */

    (FARPROC) lpfnTimerCount =
    GetProcAddress(hinstToolHelp, "TimerCount");

    if (lpfnTimerCount != NULL) {

        /* Call the TimerCount function. */

        timerinfo.dwSize = sizeof(TIMERINFO);

        if ((*lpfnTimerCount) ((TIMERINFO FAR *) &timerinfo)) {
            sprintf(szBuf, "task: %lu seconds\nVM: %lu seconds",
                timerinfo.dwmsSinceStart / 1000,
```

```
timerinfo.dwmsThisVM / 1000);
    }
    else {
        strcpy(szBuf, "TimerCount failed");
    }
    }
    else {
        strcpy(szBuf, "GetProcAddress failed");
    }

    /* Free the TOOLHELP.DLL library module. */

    FreeLibrary(hinstToolHelp);
}
else {
    strcpy(szBuf, "LoadLibrary failed");
}

MessageBox(NULL, szBuf, "Library Functions", MB_ICONHAND);
See Also
MakeProcInstance
```

## GetProfileInt (2.x)

```
UINT GetProfileInt(lpszSection, lpszEntry, default)
LPCSTR lpszSection; /* address of section */
LPCSTR lpszEntry; /* address of entry */
int default;
* return value if entry is not found
```

The GetProfileInt function retrieves the value of an integer from an entry within a specified section of the WIN.INI initialization file.

Parameter	Description
lpszSection	Points to a null-terminated string that specifies the section containing the entry.
lpszEntry	Points to the null-terminated string containing the entry whose value is to be retrieved.
default	Specifies the default value to return if the entry cannot be found. This value can be an unsigned value in the range 0 through 65,536 or a signed value in the range -32,768 through 32,768. Hexadecimal notation is accepted for both positive and negative values.

### Returns

The return value is the integer value of the string following the specified entry, if the function is successful. The return value is the value of the default parameter if the function does not find the entry. The return value is zero if the value that corresponds to the specified entry is not an integer.

### Comments

The GetProfileInt function is not case-dependent, so the strings in the lpszSection and lpszEntry parameters may contain a combination of uppercase and lowercase letters.

GetProfileInt supports hexadecimal notation. When the function is used to retrieve a negative integer, the value should be cast to an int.

An integer entry in the WIN.INI file must have the following form:

```
[section]
entry=value
.
.
.
```

If the value that corresponds to the entry consists of digits followed by nonnumeric characters, the function returns the value of the digits. For example, the function would return 102 for the line "Entry=102abc".

An application can use the [GetPrivateProfileInt](#) function to retrieve an integer from a specified file.

### Example

The following example uses the GetProfileInt function to retrieve the screen-save timeout time from the WIN.INI file:

```
WORD wTimeOut;
char szMsg[80];

wTimeOut = GetProfileInt("windows",
    "ScreenSaveTimeout", 0);

sprintf(szMsg, "timeout time is %d", wTimeOut);
MessageBox(hwnd, szMsg, "GetProfileInt", MB_OK);
```

### See Also

[GetPrivateProfileInt](#), [GetProfileString](#)



Windows 3.1 changes

The `GetProfileInt` function supports hexadecimal notation. When the `GetProfileInt` function is used to retrieve a negative integer, the value should be cast to an `int`.

## GetProfileString (2.x)

```
int GetProfileString(lpszSection, lpszEntry, lpszDefault, lpszReturnBuffer, cbReturnBuffer)
LPCSTR lpszSection;          /* address of section          */
LPCSTR lpszEntry;           /* address of entry            */
/* address of default string
/
LPCSTR lpszDefault;
/* address of default string
/
LPSTR lpszReturnBuffer;
/* address of destination buffer
/
int cbReturnBuffer;
/* size of destination buffer
/
```

The GetProfileString function retrieves the string associated with an entry within the specified section in the WIN.INI initialization file.

Parameter	Description
lpszSection	Points to a null-terminated string that specifies the section containing the entry.
lpszEntry	Points to the null-terminated string containing the entry whose associated string is to be retrieved. If this value is NULL, all entries in the section specified by the lpszSection parameter are copied to the buffer specified by the lpszReturnBuffer parameter. For more information, see the following Comments section.
lpszDefault	Points to the default value for the given entry if the entry cannot be found in the initialization file. This parameter must never be NULL.
lpszReturnBuffer	Points to the buffer that will receive the character string.
cbReturnBuffer	Specifies the size, in bytes, of the buffer pointed to by the lpszReturnBuffer parameter.

### Returns

The return value is the number of bytes copied to the buffer, not including the terminating zero, if the function is successful.

### Comments

If the lpszEntry parameter is NULL, the GetProfileString function copies all entries in the specified section to the supplied buffer. Each string will be null-terminated, with the final string terminating with two null characters. If the supplied destination buffer is too small to hold all the strings, the last string will be truncated and followed by two terminating null characters.

If the string associated with lpszEntry is enclosed in single or double quotation marks, the marks are discarded when GetProfileString returns the string.

GetProfileString is not case-dependent, so the strings in the lpszSection and lpszEntry parameters may contain a combination of uppercase and lowercase letters.

A string entry in the WIN.INI file must have the following form:

```
[section]
entry=string
.
.
.
```

An application can use the [GetPrivateProfileString](#) function to retrieve a string from a specified file.

The lpszDefault parameter must point to a valid string, even if the string is empty (its first character is zero).

### Example

The following example uses the GetProfileString function to list all the entries and strings in the [windows] section of the WIN.INI file:

```
int c, cc;
```

```

PSTR pszBuf, pszKey;
char szMsg[80], szVal[80];

/* Allocate a buffer for the entries. */
pszBuf = (PSTR) LocalAlloc(LMEM_FIXED, 1024);

/* Retrieve all the entries in the [windows] section. */
GetProfileString("windows", NULL, "", pszBuf, 1024);

/*
 * Retrieve the string for each entry, until
 * reaching the double null character.
 */

for (pszKey = pszBuf, c = 0;
    *pszKey != '\\0'; pszKey += strlen(pszKey) + 1) {

    /* Retrieve the value for each entry in the buffer. */

    GetProfileString("windows", pszKey, "not found",
        szVal, sizeof(szVal));

    cc = sprintf(szMsg, "%s = %s", pszKey, szVal);
    TextOut(hdc, 10, 15 * c++, szMsg, cc);
}

LocalFree((HANDLE) pszBuf);

```

See Also

[GetPrivateProfileString](#), [WriteProfileString](#)

## GetSelectorBase (3.1)

DWORD GetSelectorBase(uSelector)  
UINT uSelector;

The GetSelectorBase function retrieves the base address of a selector.

Parameter	Description
uSelector	Specifies the selector whose base address is retrieved.

### Returns

This function returns the base address of the specified selector.

### See Also

[GetSelectorLimit](#), [SetSelectorBase](#), [SetSelectorLimit](#)

## GetSelectorLimit (3.1)

DWORD GetSelectorLimit(uSelector)  
UINT uSelector;

The GetSelectorLimit function retrieves the limit of a selector.

Parameter	Description
uSelector	Specifies the selector whose limit is being retrieved.

### Returns

This function returns the limit of the specified selector.

### See Also

[GetSelectorBase](#), [SetSelectorBase](#), [SetSelectorLimit](#)

## GetSystemDirectory (3.0)

```
UINT GetSystemDirectory(lpszSysPath, cbSysPath)
LPSTR lpszSysPath; /* address of buffer for system directory */
UINT cbSysPath; /* size
of directory buffer
/
```

The GetSystemDirectory function retrieves the path of the Windows system directory. The system directory contains such files as Windows libraries, drivers, and font files.

Parameter	Description
lpszSysPath	Points to the buffer that is to receive the null-terminated string containing the path of the system directory.
cbSysPath	Specifies the maximum size, in bytes, of the buffer. This value should be set to at least 144 to allow sufficient room in the buffer for the path.

### Returns

The return value is the length, in bytes, of the string copied to the lpszSysPath parameter, not including the terminating null character. If the return value is greater than the size specified in the cbSysPath parameter, the return value is the size of the buffer required to hold the path. The return value is zero if the function fails.

### Comments

Applications should not create files in the system directory. If the user is running a shared version of Windows, the application will not have write access to the system directory. Applications should create files only in the directory returned by the [GetWindowsDirectory](#) function.

The path that this function retrieves does not end with a backslash unless the system directory is the root directory. For example, if the system directory is named WINDOWS\SYSTEM on drive C, the path of the system directory retrieved by this function is C:\WINDOWS\SYSTEM.

A similar function, [GetSystemDir](#), is intended for use by MS-DOS applications that set up Windows applications. Windows applications should use GetSystemDirectory, not GetSystemDir.

### Example

The following example uses the GetSystemDirectory function to determine the path of the Windows system directory:

```
WORD wReturn;
char szBuf[144];

wReturn = GetSystemDirectory((LPSTR) szBuf, sizeof(szBuf));

if (wReturn == 0)
    MessageBox(hwnd, "function failed",
        "GetSystemDirectory", MB_ICONEXCLAMATION);

else if (wReturn > sizeof(szBuf))
    MessageBox(hwnd, "buffer is too small",
        "GetSystemDirectory", MB_ICONEXCLAMATION);

else
    MessageBox(hwnd, szBuf, "GetSystemDirectory", MB_OK);
```

### See Also

[GetWindowsDirectory](#)

## GetTempDrive (2.x)

BYTE GetTempDrive(chDriveLetter)  
char chDriveLetter; /\* ignored \*/

The GetTempDrive function returns a letter that specifies a disk drive the application can use for temporary files.

Parameter	Description
chDriveLetter	This parameter is ignored.

### Returns

The return value specifies a disk drive for temporary files if the function is successful. If at least one hard disk drive is available, the function returns the letter of the first hard disk drive (usually C). If no hard disk drives are available, the function returns the letter of the current drive.

### Example

The following example uses the GetTempDrive function to determine a suitable disk drive for temporary files:

```
char szMsg[80];  
BYTE bTempDrive;  
  
bTempDrive = GetTempDrive(0);  
  
sprintf(szMsg, "temporary drive: %c", bTempDrive);  
  
MessageBox(hwnd, szMsg, "GetTempDrive", MB_OK);
```

See Also

GetTempFileName

## GetTempFileName (2.x)

```
int GetTempFileName(bDriveLetter, lpszPrefixString, uUnique, lpszTempFileName)
BYTE bDriveLetter;          /* suggested drive */
LPCSTR lpszPrefixString;
/* address of filename prefix
/
UINT uUnique;
/* number to use as prefix
/
LPSTR lpszTempFileName;
/* address of buffer for created filename
/
```

The GetTempFileName function creates a temporary filename of the following form:

drive:\path\prefixuuuu.TMP

The following list describes the filename syntax:

Element	Description
drive	Drive letter specified by the bDriveLetter parameter
path	Path of the temporary file (either the Windows directory or the directory specified in the TEMP environment variable)
prefix	All the letters (up to the first three) of the string pointed to by the lpszPrefixString parameter
uuuu	Hexadecimal value of the number specified by the uUnique parameter

Parameter	Description
bDriveLetter	Specifies the suggested drive for the temporary filename. If this parameter is zero, Windows uses the current default drive.
lpszPrefixString	Points to a null-terminated string to be used as the temporary filename prefix. This string must consist of characters in the OEM-defined character set.
uUnique	Specifies an unsigned short integer. If this parameter is nonzero, it will be appended to the temporary filename. If the parameter is zero, Windows uses the current system time to create a number to append to the filename.
lpszTempFileName	Points to the buffer that will receive the temporary filename. This string consists of characters in the OEM-defined character set. This buffer should be at least 144 bytes in length to allow sufficient room for the path.

### Returns

The return value specifies a unique numeric value used in the temporary filename. If the uUnique parameter is nonzero, the return value specifies this same number.

### Comments

Temporary files created with this function are not automatically deleted when Windows shuts down.

To avoid problems resulting from converting an OEM character string to a Windows string, an application should call the lopen function to create the temporary file.

The GetTempFileName function uses the suggested drive letter for creating the temporary filename, except in the following cases:

- If a hard disk is present, GetTempFileName always uses the drive letter of the first hard disk.
- If, however, a TEMP environment variable is defined and its value begins with a drive letter, that drive letter is used.

If the TF\_FORCEDRIVE bit of the bDriveLetter parameter is set, the preceding exceptions do not apply. The temporary filename will always be created in the current directory of the drive specified by bDriveLetter, regardless of the presence of a hard disk or the TEMP environment variable.

If the uUnique parameter is zero, GetTempFileName attempts to form a unique number based on the current system time. If a file with the resulting filename exists, the number is increased by one and the test for existence is repeated. This continues until a unique filename is found; GetTempFileName then creates a file by that name and closes it. No attempt is made to create and open the file when uUnique is nonzero.

### Example



The following example uses the `GetTempFileName` function to create a unique temporary filename on the first available hard disk:

```
HFILE hfTempFile;  
char szBuf[144];  
  
/* Create a temporary file. */  
GetTempFileName(0, "tst", 0, szBuf);  
hfTempFile = __lcreat(szBuf, 0);  
if (hfTempFile == HFILE_ERROR) {  
    ErrorHandler();  
}
```

See Also

GetTempDrive, \_\_lopen

## GetThresholdEvent (2.x)

int FAR\* GetThresholdEvent(void)

This function is obsolete. Use the Windows multimedia audio functions instead. For information about these functions, see the [Microsoft Windows Multimedia Programmer's Reference](#).

**GetThresholdStatus (2.x)**

`int GetThresholdStatus(void)`

This function is obsolete. Use the Windows multimedia audio functions instead. For information about these functions, see the [Microsoft Windows Multimedia Programmer's Reference](#).

## GetVersion (2.x)

DWORD GetVersion(void)

The GetVersion function retrieves the current version numbers of the Windows and MS-DOS operation systems.

### Returns

The return value specifies the major and minor version numbers of Windows and of MS-DOS.

### Comments

The low-order word of the return value contains the version of Windows, if the function is successful. The high-order byte contains the minor version (revision) number as a two-digit decimal number. For example, in Windows 3.1, the minor version number is 10. The low-order byte contains the major version number.

The high-order word contains the version of MS-DOS, if the function is successful. The high-order byte contains the major version; the low-order byte contains the minor version (revision) number.

### Example

The following example uses the GetVersion function to display the Windows and MS-DOS version numbers:

```
int len;
char szBuf[80];
DWORD dwVersion;

dwVersion = GetVersion();

len = sprintf(szBuf, "Windows version %d.%d\n",
    LOBYTE(LOWORD(dwVersion)),
    HIBYTE(LOWORD(dwVersion)));

sprintf(szBuf + len, "MS-DOS version %d.%d",
    HIBYTE(HIWORD(dwVersion)),
    LOBYTE(HIWORD(dwVersion)));

MessageBox(NULL, szBuf, "GetVersion", MB_ICONINFORMATION);
```

Note that the major and minor version information is reversed between the Windows version and MS-DOS version.

Win 3.1 correction

The return value is a DWORD, not a WORD. The high-order word contains the DOS version.

## GetWinDebugInfo (3.1)

```
BOOL GetWinDebugInfo(lpwdi, flags)
WINDEBDEBUGINFO FAR* lpwdi;      /* address of WINDEBDEBUGINFO structure */
UINT flags;                        /
/* flags for returned information
/
```

The GetWinDebugInfo function retrieves current system-debugging information for the debugging version of the Windows 3.1 operating system.

Parameter	Description								
lpwdi	Points to a <u>WINDEBDEBUGINFO</u> structure that is filled with debugging information.								
flags	Specifies which members of the <u>WINDEBDEBUGINFO</u> structure should be filled in. This parameter can be one or more of the following values:								
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>WDI_OPTIONS</u></td><td>Fill in the dwOptions member of <u>WINDEBDEBUGINFO</u>.</td></tr><tr><td><u>WDI_FILTER</u></td><td>Fill in the dwFilter member of <u>WINDEBDEBUGINFO</u>.</td></tr><tr><td><u>WDI_ALLOCBREAK</u></td><td>Fill in the achAllocModule, dwAllocBreak, and dwAllocCount members of <u>WINDEBDEBUGINFO</u>.</td></tr></tbody></table>	Value	Meaning	<u>WDI_OPTIONS</u>	Fill in the dwOptions member of <u>WINDEBDEBUGINFO</u> .	<u>WDI_FILTER</u>	Fill in the dwFilter member of <u>WINDEBDEBUGINFO</u> .	<u>WDI_ALLOCBREAK</u>	Fill in the achAllocModule, dwAllocBreak, and dwAllocCount members of <u>WINDEBDEBUGINFO</u> .
Value	Meaning								
<u>WDI_OPTIONS</u>	Fill in the dwOptions member of <u>WINDEBDEBUGINFO</u> .								
<u>WDI_FILTER</u>	Fill in the dwFilter member of <u>WINDEBDEBUGINFO</u> .								
<u>WDI_ALLOCBREAK</u>	Fill in the achAllocModule, dwAllocBreak, and dwAllocCount members of <u>WINDEBDEBUGINFO</u> .								

### Returns

The return value is nonzero if the function is successful. It is zero if the pointer specified in the lpwdi parameter is invalid or if the function is not called in the debugging version of Windows 3.1.

### Comments

The flags member of the returned WINDEBDEBUGINFO structure is set to the values supplied in the flags parameter of this function.

### See Also

SetWinDebugInfo, WINDEBDEBUGINFO

WDI\_OPTIONS 0x0001

Fill in the dwOptions member of WINDEBUGINFO.

WDL\_OPTIONS 0x0001



WDI\_FILTER 0x0002

Fill in the dwFilter member of WINDEBUGINFO.

WDI\_FILTER 0x0002

WDI\_ALLOCBREAK 0x0004

Fill in the achAllocModule, dwAllocBreak, and dwAllocCount members of WINDEBGINFO.

WDI\_ALLOCBREAK 0x0004

## GetWindowsDirectory (3.0)

```
UINT GetWindowsDirectory(LpszSysPath, cbSysPath)
LPSTR lpszSysPath; /* address of buffer for Windows directory */
UINT cbSysPath; /* size
of directory buffer
/
```

The GetWindowsDirectory function retrieves the path of the Windows directory. The Windows directory contains such files as Windows applications, initialization files, and help files.

Parameter	Description
lpszSysPath	Points to the buffer that will receive the null-terminated string containing the path.
cbSysPath	Specifies the maximum size, in bytes, of the buffer. This value should be set to at least 144 to allow sufficient room in the buffer for the path.

### Returns

The return value is the length, in bytes, of the string copied to the lpszSysPath parameter, not including the terminating null character. If the return value is greater than the number specified in the cbSysPath parameter, it is the size of the buffer required to hold the path. The return value is zero if the function fails.

### Comments

The Windows directory is the only directory where an application should create files. If the user is running a shared version of Windows, the Windows directory is the only directory guaranteed private to the user.

The path this function retrieves does not end with a backslash unless the Windows directory is the root directory. For example, if the Windows directory is named WINDOWS on drive C, the path retrieved by this function is C:\WINDOWS. If Windows is installed in the root directory of drive C, the path retrieved is C:\.

A similar function, GetWindowsDir, is intended for use by MS-DOS applications that set up Windows applications. Windows applications should use GetWindowsDirectory, not GetWindowsDir.

### Example

The following example uses the GetWindowsDirectory function to determine the path of the Windows directory:

```
WORD wReturn;
char szBuf[144];

wReturn = GetWindowsDirectory((LPSTR)szBuf, sizeof(szBuf));

if (wReturn == 0)
    MessageBox(hwnd, "function failed",
    "GetWindowsDirectory", MB_ICONEXCLAMATION);

else if (wReturn > sizeof(szBuf))
    MessageBox(hwnd, "buffer is too small",
    "GetWindowsDirectory", MB_ICONEXCLAMATION);

else
    MessageBox(hwnd, szBuf, "GetWindowsDirectory", MB_OK);
```

### See Also

GetSystemDirectory

## GetWinFlags (3.0)

DWORD GetWinFlags(void)

The GetWinFlags function retrieves the current Windows system and memory configuration.

### Returns

The return value specifies the current system and memory configuration.

### Comments

The configuration returned by GetWinFlags can be a combination of the following values:

Value	Meaning
<u>WF_80x87</u>	System contains an Intel math coprocessor.
<u>WF_CPU286</u>	System CPU is an 80286.
<u>WF_CPU386</u>	System CPU is an 80386.
<u>WF_CPU486</u>	System CPU is an i486.
<u>WF_ENHANCED</u>	Windows is running in 386-enhanced mode. The <u>WF_PMODE</u> flag is always set when <u>WF_ENHANCED</u> is set.
<u>WF_PAGING</u>	Windows is running on a system with paged memory.
<u>WF_PMODE</u>	Windows is running in protected mode. In Windows 3.1, this flag is always set.
<u>WF_STANDARD</u>	Windows is running in standard mode. The <u>WF_PMODE</u> flag is always set when <u>WF_STANDARD</u> is set.
<u>WF_WIN286</u>	Same as <u>WF_STANDARD</u> .
<u>WF_WIN386</u>	Same as <u>WF_ENHANCED</u> .

### Example

The following example uses the GetWinFlags function to display information about the current Windows system configuration:

```
int len;  
char szBuf[80];  
DWORD dwFlags;  
  
dwFlags = GetWinFlags();  
  
len = sprintf(szBuf, "system %s a coprocessor",  
             (dwFlags & WF_80x87) ? "contains" : "does not contain");  
TextOut(hdc, 10, 15, szBuf, len);  
  
len = sprintf(szBuf, "processor is an %s",  
             (dwFlags & WF_CPU286) ? "80286" :  
             (dwFlags & WF_CPU386) ? "80386" :  
             (dwFlags & WF_CPU486) ? "i486" : "unknown");  
TextOut(hdc, 10, 30, szBuf, len);  
  
len = sprintf(szBuf, "running in %s mode",  
             (dwFlags & WF_ENHANCED) ? "enhanced" : "standard");  
TextOut(hdc, 10, 45, szBuf, len);  
  
len = sprintf(szBuf, "%s WLO",  
             (dwFlags & WF_WLO) ? "using" : "not using");  
TextOut(hdc, 10, 60, szBuf, len);
```

WF\_80x87 0x0400

System contains an Intel math coprocessor.

WF\_80x87 0x0400



WF\_CPU286 0x0002  
System CPU is an 80286.

WF\_CPU286 0x0002

WF\_CPU386 0x0004  
System CPU is an 80386.

WF\_CPU386 0x0004

WF\_CPU486 0x0008  
System CPU is an i486.

WF\_CPU486 0x0008

WF\_ENHANCED 0x0020

Windows is running in 386-enhanced mode. The WF\_PMODE flag is always set when WF\_ENHANCED is set.

WF\_ENHANCED 0x0020



WF\_PAGING 0x0800

Windows is running on a system with paged memory.

WF\_PAGING 0x0800

WF\_PMODE 0x0001

Windows is running in protected mode. In Windows 3.1, this flag is always set.

WF\_PMODE 0x0001

WF\_STANDARD 0x0010

Windows is running in standard mode. The WF\_PMODE flag is always set when WF\_STANDARD is set.

WF\_STANDARD 0x0010

WF\_WIN286 0x0010  
Same as WF\_STANDARD.

WF\_WIN286 0x0010



WF\_WIN386 0x0020  
Same as WF\_ENHANCED.

WF\_WIN386 0x0020

## GlobalAlloc (2.x)

```
HGLOBAL GlobalAlloc(fuAlloc, cbAlloc)
UINT fuAlloc;          /* how to allocate object */
DWORD cbAlloc;        /* size of object */
```

The GlobalAlloc function allocates the specified number of bytes from the global heap.

Parameter	Description																												
fuAlloc	Specifies how to allocate memory. This parameter can be a combination of the following values:																												
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>GHND</u></td><td>Combines the <u>GMEM_MOVEABLE</u> and <u>GMEM_ZEROINIT</u> flags.</td></tr><tr><td><u>GMEM_DDESHARE</u></td><td>Allocates sharable memory. This flag is used for dynamic data exchange (DDE) only. This flag is equivalent to <u>GMEM_SHARE</u>.</td></tr><tr><td><u>GMEM_DISCARDABLE</u></td><td>Allocates discardable memory. This flag can only be used with the <u>GMEM_MOVEABLE</u> flag.</td></tr><tr><td><u>GMEM_FIXED</u></td><td>Allocates fixed memory. The <u>GMEM_FIXED</u> and <u>GMEM_MOVEABLE</u> flags cannot be combined.</td></tr><tr><td><u>GMEM_LOWER</u></td><td>Same as <u>GMEM_NOT_BANKED</u>. This flag is ignored in Windows 3.1.</td></tr><tr><td><u>GMEM_MOVEABLE</u></td><td>Allocates movable memory. The <u>GMEM_FIXED</u> and <u>GMEM_MOVEABLE</u> flags cannot be combined.</td></tr><tr><td><u>GMEM_NOCOMPACT</u></td><td>Does not compact or discard memory to satisfy the allocation request.</td></tr><tr><td><u>GMEM_NODISCARD</u></td><td>Does not discard memory to satisfy the allocation request.</td></tr><tr><td><u>GMEM_NOT_BANKED</u></td><td>Allocates non-banked memory (memory is not within the memory provided by expanded memory). This flag cannot be used with the <u>GMEM_NOTIFY</u> flag. This flag is ignored in Windows 3.1.</td></tr><tr><td><u>GMEM_NOTIFY</u></td><td>Calls the notification routine if the memory object is discarded.</td></tr><tr><td><u>GMEM_SHARE</u></td><td>Allocates memory that can be shared with other applications. This flag is equivalent to <u>GMEM_DDESHARE</u>.</td></tr><tr><td><u>GMEM_ZEROINIT</u></td><td>Initializes memory contents to zero.</td></tr><tr><td><u>GPTR</u></td><td>Combines the <u>GMEM_FIXED</u> and <u>GMEM_ZEROINIT</u> flags.</td></tr></tbody></table>	Value	Meaning	<u>GHND</u>	Combines the <u>GMEM_MOVEABLE</u> and <u>GMEM_ZEROINIT</u> flags.	<u>GMEM_DDESHARE</u>	Allocates sharable memory. This flag is used for dynamic data exchange (DDE) only. This flag is equivalent to <u>GMEM_SHARE</u> .	<u>GMEM_DISCARDABLE</u>	Allocates discardable memory. This flag can only be used with the <u>GMEM_MOVEABLE</u> flag.	<u>GMEM_FIXED</u>	Allocates fixed memory. The <u>GMEM_FIXED</u> and <u>GMEM_MOVEABLE</u> flags cannot be combined.	<u>GMEM_LOWER</u>	Same as <u>GMEM_NOT_BANKED</u> . This flag is ignored in Windows 3.1.	<u>GMEM_MOVEABLE</u>	Allocates movable memory. The <u>GMEM_FIXED</u> and <u>GMEM_MOVEABLE</u> flags cannot be combined.	<u>GMEM_NOCOMPACT</u>	Does not compact or discard memory to satisfy the allocation request.	<u>GMEM_NODISCARD</u>	Does not discard memory to satisfy the allocation request.	<u>GMEM_NOT_BANKED</u>	Allocates non-banked memory (memory is not within the memory provided by expanded memory). This flag cannot be used with the <u>GMEM_NOTIFY</u> flag. This flag is ignored in Windows 3.1.	<u>GMEM_NOTIFY</u>	Calls the notification routine if the memory object is discarded.	<u>GMEM_SHARE</u>	Allocates memory that can be shared with other applications. This flag is equivalent to <u>GMEM_DDESHARE</u> .	<u>GMEM_ZEROINIT</u>	Initializes memory contents to zero.	<u>GPTR</u>	Combines the <u>GMEM_FIXED</u> and <u>GMEM_ZEROINIT</u> flags.
Value	Meaning																												
<u>GHND</u>	Combines the <u>GMEM_MOVEABLE</u> and <u>GMEM_ZEROINIT</u> flags.																												
<u>GMEM_DDESHARE</u>	Allocates sharable memory. This flag is used for dynamic data exchange (DDE) only. This flag is equivalent to <u>GMEM_SHARE</u> .																												
<u>GMEM_DISCARDABLE</u>	Allocates discardable memory. This flag can only be used with the <u>GMEM_MOVEABLE</u> flag.																												
<u>GMEM_FIXED</u>	Allocates fixed memory. The <u>GMEM_FIXED</u> and <u>GMEM_MOVEABLE</u> flags cannot be combined.																												
<u>GMEM_LOWER</u>	Same as <u>GMEM_NOT_BANKED</u> . This flag is ignored in Windows 3.1.																												
<u>GMEM_MOVEABLE</u>	Allocates movable memory. The <u>GMEM_FIXED</u> and <u>GMEM_MOVEABLE</u> flags cannot be combined.																												
<u>GMEM_NOCOMPACT</u>	Does not compact or discard memory to satisfy the allocation request.																												
<u>GMEM_NODISCARD</u>	Does not discard memory to satisfy the allocation request.																												
<u>GMEM_NOT_BANKED</u>	Allocates non-banked memory (memory is not within the memory provided by expanded memory). This flag cannot be used with the <u>GMEM_NOTIFY</u> flag. This flag is ignored in Windows 3.1.																												
<u>GMEM_NOTIFY</u>	Calls the notification routine if the memory object is discarded.																												
<u>GMEM_SHARE</u>	Allocates memory that can be shared with other applications. This flag is equivalent to <u>GMEM_DDESHARE</u> .																												
<u>GMEM_ZEROINIT</u>	Initializes memory contents to zero.																												
<u>GPTR</u>	Combines the <u>GMEM_FIXED</u> and <u>GMEM_ZEROINIT</u> flags.																												
cbAlloc	Specifies the number of bytes to be allocated.																												

### Returns

The return value is the handle of the newly allocated global memory object, if the function is successful. Otherwise, it is NULL.

### Comments

To convert the handle returned by the GlobalAlloc function into a pointer, an application should use the GlobalLock function.

If this function is successful, it allocates at least the amount requested. If the amount allocated is greater than the amount requested, the application can use the entire amount. To determine the size of a global memory object, an application can use the GlobalSize function.

To free a global memory object, an application should use the GlobalFree function. To change the size or attributes of an allocated memory object, an application can use the GlobalReAlloc function.

The largest memory object that an application can allocate on an 80286 processor is 1 megabyte less 80 bytes. The largest block on an 80386 processor is 16 megabytes less 64K.

If the cbAlloc parameter is zero, the GlobalAlloc function returns a handle of a memory object that is marked as discarded.

### Example

The following example uses the `GlobalAlloc` and `GlobalLock` functions to allocate memory, and then calls the `GlobalUnlock` and `GlobalFree` functions to free it.

```
HGLOBAL hglb;  
void FAR* lpvBuffer;  
  
hglb = GlobalAlloc(GPTR, 1024);  
lpvBuffer = GlobalLock(hglb);  
.  
.  
GlobalUnlock(hglb);  
GlobalFree(hglb);
```

See Also

`GlobalFree`, `GlobalLock`, `GlobalNotify`, `GlobalReAlloc`, `GlobalSize`, `GlobalUnlock`, `LocalAlloc`

GHND (GMEM\_MOVEABLE | GMEM\_ZEROINIT)

Combines the GMEM\_MOVEABLE and GMEM\_ZEROINIT flags.

GHND (GMEM\_MOVEABLE | GMEM\_ZEROINIT)

GMEM\_DDESHARE 0x2000

Allocates sharable memory. This flag is used for dynamic data exchange (DDE) only. This flag is equivalent to GMEM\_SHARE.

GMEM\_DDESHARE 0x2000



GMEM\_DISCARDABLE 0x0100

Allocates discardable memory. This flag can only be used with the GMEM\_MOVEABLE flag.

GMEM\_DISCARDABLE 0x0100

GMEM\_FIXED 0x0000

Allocates fixed memory. The GMEM\_FIXED and GMEM\_MOVEABLE flags cannot be combined.

GMEM\_FIXED 0x0000

GMEM\_LOWER GMEM\_NOT\_BANKED

Same as GMEM\_NOT\_BANKED. This flag is ignored in Windows 3.1.

GMEM\_LOWER GMEM\_NOT\_BANKED

GMEM\_MOVEABLE 0x0002

Allocates movable memory. The GMEM\_FIXED and GMEM\_MOVEABLE flags cannot be combined.

GMEM\_MOVEABLE 0x0002



GMEM\_NOCOMPACT 0x0010

Does not compact or discard memory to satisfy the allocation request.

GMEM\_NOCOMPACT 0x0010

GMEM\_NODISCARD 0x0020

Does not discard memory to satisfy the allocation request.

GMEM\_NODISCARD 0x0020

GMEM\_NOT\_BANKED 0x1000

Allocates non-banked memory (memory is not within the memory provided by expanded memory). This flag cannot be used with the GMEM\_NOTIFY flag. This flag is ignored in Windows 3.1.

GMEM\_NOT\_BANKED 0x1000

GMEM\_NOTIFY 0x4000

Calls the notification routine if the memory object is discarded.

GMEM\_NOTIFY 0x4000



GMEM\_SHARE 0x2000

Allocates memory that can be shared with other applications. This flag is equivalent to GMEM\_DDESHARE.

GMEM\_SHARE 0x2000

GMEM\_ZEROINIT 0x0040

Initializes memory contents to zero.

GMEM\_ZEROINIT 0x0040

GPTR (GMEM\_FIXED | GMEM\_ZEROINIT)

Combines the GMEM\_FIXED and GMEM\_ZEROINIT flags.

GPTR (GMEM\_FIXED | GMEM\_ZEROINIT)

## GlobalCompact (2.x)

```
DWORD GlobalCompact(dwMinFree)
DWORD dwMinFree; /* amount of memory requested */
```

The GlobalCompact function rearranges memory currently allocated to the global heap so that the specified amount of memory is free. If the function cannot free the requested amount of memory, it frees as much as possible.

Parameter	Description
dwMinFree	Specifies the number of contiguous free bytes desired. If this parameter is zero, the function does not discard memory, but the return value is valid.

### Returns

The return value specifies the number of bytes in the largest free global memory object in the global heap. If the dwMinFree parameter is zero, the return value specifies the number of bytes in the largest free object that Windows can generate if it removes all discardable objects.

### Comments

If an application passes the return value to the GlobalAlloc function, the GMEM\_NOCOMPACT or GMEM\_NODISCARD flag should not be used.

This function always rearranges movable memory objects before checking for free memory. Then it checks the memory currently allocated to the global heap for the number of contiguous free bytes specified by the dwMinFree parameter. If the specified amount of memory is not available, the function discards unlocked discardable objects, until the requested space is generated (if possible).

This function is not very useful in most enhanced-mode configurations, which rely on virtual memory. In such an environment, an application can discover how much memory is available by requesting the memory and checking the error value.

### See Also

GlobalAlloc

## GlobalDosAlloc (3.0)

```
DWORD GlobalDosAlloc(cbAlloc)
DWORD cbAlloc; /* number of bytes to allocate */
```

The GlobalDosAlloc function allocates global memory that can be accessed by MS-DOS running in real mode. The memory is guaranteed to exist in the first megabyte of linear address space.

An application should not use this function unless it is absolutely necessary, because the memory pool from which the object is allocated is a scarce system resource.

Parameter	Description
cbAlloc	Specifies the number of bytes to be allocated.

### Returns

The return value contains a paragraph-segment value in its high-order word and a selector in its low-order word. An application can use the paragraph-segment value to access memory in real mode and the selector to access memory in protected mode. If Windows cannot allocate a block of memory of the requested size, the return value is zero.

### Comments

Memory allocated by using the GlobalDosAlloc function does not need to be locked by using the GlobalLock function.

### See Also

GlobalDosFree



## GlobalDosFree (3.0)

```
UINT GlobalDosFree(uSelector)  
UINT uSelector;          /* memory to free */
```

The GlobalDosFree function frees a global memory object previously allocated by the GlobalDosAlloc function.

Parameter	Description
uSelector	Identifies the memory object to be freed.

### Returns

The return value is zero if the function is successful. Otherwise, it is equal to the uSelector parameter.

See Also

GlobalDosAlloc

## GlobalFix (3.0)

```
void GlobalFix(hglb)
HGLOBAL hglb;      /* handle of object to fix */
```

The GlobalFix function prevents the given global memory object from moving in linear memory.

This function interferes with effective Windows memory management and can result in linear-address fragmentation. Few applications need to fix memory in linear address space.

Parameter	Description
hglb	Identifies the global memory object to be fixed in linear memory.

### Returns

This function does not return a value.

### Comments

The object is locked into linear memory at its current address, and its lock count is incremented (increased by one). Locked memory is not subject to moving or discarding except when the memory object is being reallocated by the GlobalReAlloc function. The object remains locked in memory until its lock count is decreased to zero.

Each time an application calls the GlobalFix function for a memory object, it must eventually call the GlobalUnfix function, which decrements (decreases by one) the lock count for the object. Other functions also can affect the lock count of a memory object. For a list of these functions, see the description of the GlobalFlags function.

### See Also

GlobalFlags, GlobalReAlloc, GlobalUnfix

## GlobalFlags (2.x)

```
UINT GlobalFlags(hglb)
HGLOBAL hglb; /* handle of global memory object */
```

The GlobalFlags function returns information about the given global memory object.

Parameter	Description
hglb	Identifies the global memory object.

### Returns

The return value specifies the memory-allocation flag and the lock count for the memory object, if the function is successful.

### Comments

When an application masks out the lock count in the low-order byte of the return value, the return value contains one of the following allocation flags:

Value	Meaning
GMEM_DISCARDABLE	Object can be discarded.
GMEM_DISCARDED	Object has been discarded.

The low-order byte of the return value contains the lock count of the object. Use the GMEM\_LOCKCOUNT mask to retrieve the lock count from the return value.

The following functions can affect the lock count of a global memory object:

Increments lock count	Decrements lock count
<u>GlobalFix</u>	<u>GlobalUnfix</u>
<u>GlobalLock</u>	<u>GlobalUnlock</u>

### See Also

GlobalFix, GlobalLock, GlobalUnfix, GlobalUnlock

## GlobalFree (2.x)

```
HGLOBAL GlobalFree(hglb)
HGLOBAL hglb; /* handle of object to free */
```

The GlobalFree function frees the given global memory object (if the object is not locked) and invalidates its handle.

Parameter	Description
hglb	Identifies the global memory object to be freed.

### Returns

The return value is NULL if the function is successful. Otherwise, it is equal to the hglb parameter.

### Comments

The GlobalFree function cannot be used to free a locked memory object--that is, a memory object with a lock count greater than zero. For a list of the functions that affect the lock count, see the description of the [GlobalFlags](#) function.

Once freed, the handle of the memory object must not be used again. Attempting to free the same memory object more than once can cause Windows to terminate abnormally.

### Example

The following code fragment uses the [GlobalAlloc](#) and [GlobalLock](#) functions to allocate memory, and then calls the [GlobalUnlock](#) and [GlobalFree](#) functions to free it.

```
HGLOBAL hglb;
void FAR* lpvBuffer;

hglb = GlobalAlloc(GPTR, 1024);
lpvBuffer = GlobalLock(hglb);
.
.
GlobalUnlock(hglb);
GlobalFree(hglb);
```

### See Also

[GlobalDiscard](#), [GlobalFlags](#), [GlobalLock](#), [GlobalUnlock](#)

## GlobalHandle (2.x)

DWORD GlobalHandle(uGlobalSel)

UINT uGlobalSel; /\* selector of global memory object \*/

The GlobalHandle function retrieves the handle of the specified global memory object.

Parameter	Description
uGlobalSel	Specifies the selector of a global memory object.

### Returns

The low-order word of the return value contains the handle of the global memory object, and the high-order word contains the selector of the memory object, if the function is successful. The return value is NULL if no handle exists for the memory object.

## GlobalLock (2.x)

```
void FAR* GlobalLock(hglb)
HGLOBAL hglb;      /* handle of memory object to lock */
```

The GlobalLock function returns a pointer to the given global memory object. GlobalLock increments (increases by one) the lock count of movable objects and locks the memory. Locked memory will not be moved or discarded unless the memory object is reallocated by the GlobalReAlloc function. The object remains locked in memory until its lock count is decreased to zero.

Parameter	Description
hglb	Identifies the global memory object to be locked.

### Returns

The return value points to the first byte of memory in the global object, if the function is successful. It is NULL if the object has been discarded or an error occurs.

### Comments

Each time an application calls the GlobalLock function for an object, it must eventually call the GlobalUnlock function for the object.

This function will return NULL if an application attempts to lock a memory object with a zero-byte size.

If GlobalLock incremented the lock count for the object, GlobalUnlock decrements the lock count for the object. Other functions can also affect the lock count of a memory object. For a list of these functions, see the description of the GetGlobalFlags function.

Discarded objects always have a lock count of zero.

### Example

The following example uses the GlobalAlloc and GlobalLock functions to allocate memory, and then calls the GlobalUnlock and GlobalFree functions to free it.

```
HGLOBAL hglb;
void FAR* lpvBuffer;

hglb = GlobalAlloc(GPTR, 1024);
lpvBuffer = GlobalLock(hglb);
.
.
.
GlobalUnlock(hglb);
GlobalFree(hglb);
```

### See Also

GlobalAlloc, GlobalFlags, GlobalFree, GlobalLock, GlobalReAlloc, GlobalUnlock

## GlobalLRUNewest (2.x)

```
HGLOBAL GlobalLRUNewest(hglb)
HGLOBAL hglb;      /* handle of memory object to move */
```

The GlobalLRUNewest function moves a global memory object to the newest least-recently-used (LRU) position in memory. This greatly reduces the likelihood that the object will be discarded soon, but does not prevent the object from eventually being discarded.

Parameter	Description
hglb	Identifies the global memory object to be moved.

### Returns

The return value is NULL if the hglb parameter is not a valid handle.

### Comments

The GlobalLRUNewest function is useful only if the given object is discardable.

### See Also

[GlobalLRUOldest](#)

## GlobalLRUOldest (2.x)

```
HGLOBAL GlobalLRUOldest(hglb)
HGLOBAL hglb;      /* handle of memory object to move */
```

The GlobalLRUOldest function moves a global memory object to the oldest least-recently-used (LRU) position in memory. This makes the memory object the next candidate for discarding.

Parameter	Description
hglb	Identifies the global memory object to be moved.

### Returns

The return value is NULL if the hglb parameter does not identify a valid handle.

### Comments

The GlobalLRUOldest function is useful only if the hglb parameter is discardable.

### See Also

GlobalLRUNewest



## GlobalNotify (2.x)

```
void GlobalNotify(lpNotifyProc)
GNOTIFYPROC lpNotifyProc; /* instance address of callback function */
```

The GlobalNotify function installs a notification procedure for the current task. A notification procedure is a library-defined callback function that the system calls whenever a global memory object allocated with the GMEM\_NOTIFY flag is about to be discarded.

Parameter	Description
lpNotifyProc	Specifies the address of the current task's notification procedure. For more information, see the description of the <u>NotifyProc</u> callback function.

### Returns

This function does not return a value.

### Comments

An application must not call the GlobalNotify function more than once per instance.

The system does not call the notification procedure when discarding memory that belongs to a dynamic-link library (DLL).

If the object is discarded, the application must use the GMEM\_NOTIFY flag when it calls the GlobalReAlloc function to recreate the object. Otherwise, the application will not be notified when the object is discarded again.

If the notification procedure returns a nonzero value, Windows discards the global memory object. If the procedure returns zero, the block is not discarded.

The address of the NotifyProc callback function (specified in the lpNotifyProc parameter) must be in a fixed code segment of a dynamic-link library.

### See Also

GlobalReAlloc, NotifyProc

## GlobalPageLock (3.0)

```
UINT GlobalPageLock(hglb)
HGLOBAL hglb; /* selector of global memory to lock */
```

The GlobalPageLock function increments (increases by one) the page-lock count for the memory associated with the given global selector. As long as its page-lock count is nonzero, the data that the selector references is guaranteed to remain in memory at the same physical address.

Parameter	Description
hglb	Specifies the selector of the memory to be page-locked.

### Returns

The return value specifies the page-lock count after the function has incremented it. If the function fails, the return value is zero.

### Comments

Because using this function violates preferred Windows programming practices, an application should not use it unless absolutely necessary. The function is intended to be used for dynamically allocated data that must be accessed at interrupt time. For this reason, it must be called only from a dynamic-link library (DLL).

The GlobalPageLock function increments the page-lock count for the block of memory, and the GlobalPageUnlock function decrements (decreases by one) the page-lock count. Page-locking operations can be nested, but each page-locking must be balanced by a corresponding unlocking.

### See Also

GlobalPageUnlock

## GlobalPageUnlock (3.0)

```
UINT GlobalPageUnlock(hglb)
HGLOBAL hglb; /* selector of global memory to unlock */
```

The GlobalPageLock function decrements (decreases by one) the page-lock count for the memory associated with the specified global selector. When the page-lock count reaches zero, the data that the selector references is no longer guaranteed to remain in memory at the same physical address.

Parameter	Description
hglb	Specifies the selector of the memory to be page-unlocked.

### Returns

The return value specifies the page-lock count after the function has decremented it. If the function fails, the return value is zero.

### Comments

Because using this function violates preferred Windows programming practices, an application should not use it unless absolutely necessary. The function is intended to be used for dynamically allocated data that must be accessed at interrupt time. For this reason, it must only be called from a dynamic-link library (DLL).

The GlobalPageLock function increments the page-lock count for the block of memory, and the GlobalPageUnlock function decrements the page-lock count. Page-locking operations can be nested, but each page-locking must be balanced by a corresponding unlocking.

### See Also

[GlobalPageLock](#)

## GlobalReAlloc (2.x)

```
HGLOBAL GlobalReAlloc(hglb, cbNewSize, fuAlloc)
HGLOBAL hglb;          /* handle of memory object to reallocate */
DWORD cbNewSize;      /* new size of object */
/
UINT fuAlloc;         /* how object is reallocated */
/
```

The GlobalReAlloc function changes the size or attributes of the given global memory object.

Parameter	Description												
hglb	Identifies the global memory object to be reallocated.												
cbNewSize	Specifies the new size of the memory object.												
fuAlloc	Specifies how to reallocate the global object. If this parameter includes GMEM_MODIFY, the GlobalReAlloc function ignores the cbNewSize parameter.												
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>GMEM_DISCARDABLE</td><td>Causes a previously movable object to become discardable. This flag can be used only with GMEM_MODIFY.</td></tr><tr><td>GMEM_MODIFY</td><td>Modifies the object's memory flags. This flag can be used with GMEM_DISCARDABLE and GMEM_MOVEABLE.</td></tr><tr><td>GMEM_MOVEABLE</td><td>Causes a previously movable and discardable object to be discarded, if the cbNewSize parameter is zero and the object's lock count is zero. If cbNewSize is zero and the object is not movable and discardable, this flag causes the GlobalReAlloc function to fail. If the cbNewSize parameter is nonzero and the object identified by the hglb parameter is fixed, this flag allows the reallocated object to be moved to a new fixed location. If a movable object is locked, this flag allows the object to be moved to a new locked location without invalidating the handle. This may occur even if the object is currently locked by a previous call to the GlobalLock function. If this flag is used with GMEM_MODIFY, the GlobalReAlloc function changes a fixed memory object to a movable memory object.</td></tr><tr><td>GMEM_NODISCARD</td><td>Prevents memory from being discarded to satisfy the allocation request. This flag cannot be used with GMEM_MODIFY.</td></tr><tr><td>GMEM_ZEROINIT</td><td>Causes the additional memory to be initialized to zero if the object is growing. This flag cannot be used with GMEM_MODIFY.</td></tr></tbody></table>	Value	Meaning	GMEM_DISCARDABLE	Causes a previously movable object to become discardable. This flag can be used only with GMEM_MODIFY.	GMEM_MODIFY	Modifies the object's memory flags. This flag can be used with GMEM_DISCARDABLE and GMEM_MOVEABLE.	GMEM_MOVEABLE	Causes a previously movable and discardable object to be discarded, if the cbNewSize parameter is zero and the object's lock count is zero. If cbNewSize is zero and the object is not movable and discardable, this flag causes the GlobalReAlloc function to fail. If the cbNewSize parameter is nonzero and the object identified by the hglb parameter is fixed, this flag allows the reallocated object to be moved to a new fixed location. If a movable object is locked, this flag allows the object to be moved to a new locked location without invalidating the handle. This may occur even if the object is currently locked by a previous call to the GlobalLock function. If this flag is used with GMEM_MODIFY, the GlobalReAlloc function changes a fixed memory object to a movable memory object.	GMEM_NODISCARD	Prevents memory from being discarded to satisfy the allocation request. This flag cannot be used with GMEM_MODIFY.	GMEM_ZEROINIT	Causes the additional memory to be initialized to zero if the object is growing. This flag cannot be used with GMEM_MODIFY.
Value	Meaning												
GMEM_DISCARDABLE	Causes a previously movable object to become discardable. This flag can be used only with GMEM_MODIFY.												
GMEM_MODIFY	Modifies the object's memory flags. This flag can be used with GMEM_DISCARDABLE and GMEM_MOVEABLE.												
GMEM_MOVEABLE	Causes a previously movable and discardable object to be discarded, if the cbNewSize parameter is zero and the object's lock count is zero. If cbNewSize is zero and the object is not movable and discardable, this flag causes the GlobalReAlloc function to fail. If the cbNewSize parameter is nonzero and the object identified by the hglb parameter is fixed, this flag allows the reallocated object to be moved to a new fixed location. If a movable object is locked, this flag allows the object to be moved to a new locked location without invalidating the handle. This may occur even if the object is currently locked by a previous call to the GlobalLock function. If this flag is used with GMEM_MODIFY, the GlobalReAlloc function changes a fixed memory object to a movable memory object.												
GMEM_NODISCARD	Prevents memory from being discarded to satisfy the allocation request. This flag cannot be used with GMEM_MODIFY.												
GMEM_ZEROINIT	Causes the additional memory to be initialized to zero if the object is growing. This flag cannot be used with GMEM_MODIFY.												

### Returns

The return value is the handle of the reallocated global memory if the function is successful. It is NULL if the object cannot be reallocated as specified.

### Comments

If GlobalReAlloc reallocates a movable object, the return value is a handle to the memory. To access the memory, an application must use the GlobalLock function to convert the handle to a pointer.

To free a global memory object, an application should use the GlobalFree function.

The GMEM\_ZEROINIT flag will cause applications to fail if it is used as shown in the following sequence:

```
hMem = GlobalAlloc(GMEM_ZEROINIT | (other flags), dwSize1);
.
.
hMem = GlobalReAlloc(hMem, dwSize2, GMEM_ZEROINIT | (other flags));
/* where dwSize2 > dwSize1. */
.
.
hMem = GlobalReAlloc(hMem, dwSize3, GMEM_ZEROINIT | (other flags));
/* where dwSize3 < dwSize2. */
.
.
hMem = GlobalReAlloc(hMem, dwSize4, GMEM_ZEROINIT | (other flags));
/* GMEM_ZEROINIT fails when dwSize4 > dwSize3. */
```

In the last step of the preceding example, the memory between dwSize3 and the internal allocation boundary is not set to zero. After the last step, the contents of the buffer equal its contents prior to the call to GlobalReAlloc that specified dwSize3.

See Also

[GlobalAlloc](#), [GlobalDiscard](#), [GlobalFree](#), [GlobalLock](#)

## GlobalSize (2.x)

DWORD GlobalSize(hglb)

HGLOBAL hglb; /\* handle of memory object to return size of \*/

The GlobalSize function retrieves the current size, in bytes, of the given global memory object.

Parameter	Description
hglb	Identifies the global memory object.

### Returns

The return value specifies the size, in bytes, of the memory object. It is zero if the specified handle is not valid or if the object has been discarded.

### Comments

The size of a memory object is sometimes larger than the size requested at the time the memory was allocated.

An application should call the GlobalFlags function prior to calling the GlobalSize function, to verify that the specified memory object was not discarded. If the memory object has been discarded, the return value for GlobalSize is meaningless.

### See Also

GlobalAlloc, GlobalFlags

## GlobalUnfix (3.0)

```
void GlobalUnfix(hglb)
HGLOBAL hglb; /* handle of global memory to unlock */
```

The GlobalUnfix function cancels the effects of the GlobalFix function and allows a global memory object to be moved in linear memory.

Parameter	Description
hglb	Identifies the global memory object to be unlocked.

### Returns

This function does not return a value.

### Comments

This function interferes with effective Windows memory management and can result in linear-address fragmentation. Few applications need to fix memory in linear address space.

Each time an application calls the GlobalFix function for an object, it must eventually call the GlobalUnfix function for the object.

GlobalUnfix decrements (decreases by one) the object's lock count and returns the new lock count in the CX register. The object is completely unlocked and subject to moving or discarding if the lock count is decremented to zero. Other functions also can affect the lock count of a memory object. For a list of these functions, see the description of the GlobalFlags function.

### See Also

GlobalFix, GlobalFlags

## GlobalUnlock (2.x)

```
BOOL GlobalUnlock(hglb)
HGLOBAL hglb; /* handle of global memory to unlock */
```

The GlobalUnlock function unlocks the given global memory object. This function has no effect on fixed memory.

Parameter	Description
hglb	Identifies the global memory object to be unlocked.

### Returns

The return value is zero if the object's lock count was decremented (decreased by one) to zero. Otherwise, the return value is nonzero.

### Comments

With movable or discardable memory, this function decrements the object's lock count. The object is completely unlocked and subject to moving or discarding if the lock count is decreased to zero.

This function returns nonzero if the given memory object is not movable. An application should not rely on the return value to determine the number of times it must subsequently call the GlobalUnlock function for the memory object.

Other functions can also affect the lock count of a memory object. For a list of the functions that affect the lock count, see the description of the [GlobalFlags](#) function.

Each time an application calls [GlobalLock](#) for an object, it must eventually call the GlobalUnlock function for the object.

### Example

The following example uses the [GlobalAlloc](#) and [GlobalLock](#) functions to allocate memory, and then calls the GlobalUnlock and [GlobalFree](#) functions to free it.

```
HGLOBAL hglb;
void FAR* lpvBuffer;

hglb = GlobalAlloc(GPTR, 1024);
lpvBuffer = GlobalLock(hglb);
.
.
.
GlobalUnlock(hglb);
GlobalFree(hglb);
```

### See Also

[GlobalAlloc](#), [GlobalFlags](#), [GlobalFree](#), [GlobalLock](#), GlobalUnlock, [UnlockResource](#)



GlobalUnWire (2.x)

BOOL GlobalUnWire(hglb)  
HGLOBAL hglb;

This function should not be used in Windows 3.1.

See Also  
GlobalUnlock

## GlobalWire (2.x)

```
void FAR* GlobalWire(hglb)  
HGLOBAL hglb;
```

This function should not be used in Windows 3.1.

See Also

[GlobalLock](#)

### hmemcpy (3.1)

```
void hmemcpy(hpvDest, hpvSource, cbCopy)
void _huge* hpvDest;          /* address of destination buffer */
const void _huge* hpvSource; /* address of source buffer */
/
long cbCopy;
* number of bytes to copy
/
```

The hmemcpy function copies bytes from a source buffer to a destination buffer. This function supports huge memory objects (that is, objects larger than 64K, allocated using the GlobalAlloc function).

Parameter	Description
hpvDest	Points to a buffer that receives the copied bytes.
hpvSource	Points to a buffer that contains the bytes to be copied.
cbCopy	Specifies the number of bytes to be copied.

#### Returns

This function does not return a value.

#### Comments

The result of the hmemcpy function is undefined if the buffers identified by hpvDest and hpvSource overlap.

#### See Also

hread, hwrite, lstrcpy

## InitAtomTable (2.x)

```
BOOL InitAtomTable(cTableEntries)
int cTableEntries;      /* size of atom table */
```

The InitAtomTable function initializes the local atom hash table and sets it to the specified size.

An application need not use this function to use a local atom table. The default size of the local and global atom hash tables is 37 table entries. If an application uses InitAtomTable, however, it should call the function before any other atom-management function.

Parameter	Description
cTableEntries	Specifies the size, in table entries, of the atom hash table. This value should be a prime number.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

If an application uses a large number of local atoms, it can increase the size of the local atom table, reducing the time required to add an atom to the local atom table or to find an atom in the table. However, this increases the amount of memory required to maintain the table.

The size of the global atom table cannot be changed from its default size of 37 entries.

### Example

The following example uses the InitAtomTable function to change the size of the local atom table to 73:

```
BOOL fSuccess;

fSuccess = InitAtomTable(73);

if (fSuccess)
    MessageBox(hwnd, "table initialization succeeded",
    "InitAtomTable", MB_OK);
else
    MessageBox(hwnd, "table initialization failed",
    "InitAtomTable", MB_ICONEXCLAMATION);
```

### IsBadCodePtr (3.1)

```
BOOL IsBadCodePtr(lpfn)
FARPROC lpfn;          /* pointer to test */
```

The IsBadCodePtr function determines whether a pointer to executable code is valid.

Parameter	Description
lpfn	Points to a function.

#### Returns

The return value is nonzero if the pointer is bad (that is, if it does not point to executable code). The return value is zero if the pointer is good.

#### See Also

[IsBadHugeReadPtr](#), [IsBadHugeWritePtr](#), [IsBadReadPtr](#), [IsBadStringPtr](#), [IsBadWritePtr](#)

### IsBadHugeReadPtr (3.1)

```
BOOL IsBadHugeReadPtr(lp, cb)
const void _huge* lp;          /* pointer to test          */
DWORD cb;                      /* number of allocated      */
bytes
/
```

\*

The IsBadHugeReadPtr function determines whether a huge pointer to readable memory is valid.

Parameter	Description
lp	Points to the beginning of a block of allocated memory. The data object may reside anywhere in memory and may exceed 64K in size.
cb	Specifies the number of bytes of memory that were allocated.

#### Returns

The return value is nonzero if the pointer is bad (that is, if it does not point to readable memory of the specified size). The return value is zero if the pointer is good.

#### See Also

IsBadCodePtr, IsBadHugeWritePtr, IsBadReadPtr, IsBadStringPtr, IsBadWritePtr

### IsBadHugeWritePtr (3.1)

```
BOOL IsBadHugeWritePtr(lp, cb)
void _huge* lp;          /* pointer to test          */
DWORD cb;                /* number of allocated bytes  */
/
```

The IsBadHugeWritePtr function determines whether a huge pointer to writable memory is valid.

Parameter	Description
lp	Points to the beginning of a block of allocated memory. The data object may reside anywhere in memory and may exceed 64K in size.
cb	Specifies the number of bytes of memory that were allocated.

#### Returns

The return value is nonzero if the pointer is bad (that is, if it does not point to writable memory of the specified size). The return value is zero if the pointer is good.

#### See Also

IsBadCodePtr, IsBadHugeReadPtr, IsBadReadPtr, IsBadStringPtr, IsBadWritePtr

### IsBadReadPtr (3.1)

```
BOOL IsBadReadPtr(lp, cb)
const void FAR* lp;          /* pointer to test          */
UINT cb;                    /* number of allocated     */
bytes                       *
/
```

The IsBadReadPtr function determines whether a pointer to readable memory is valid.

Parameter	Description
lp	Points to the beginning of a block of allocated memory.
cb	Specifies the number of bytes of memory that were allocated.

#### Returns

The return value is nonzero if the pointer is bad (that is, if it does not point to readable memory of the specified size). The return value is zero if the pointer is good.

#### See Also

IsBadCodePtr, IsBadHugeReadPtr, IsBadHugeWritePtr, IsBadStringPtr, IsBadWritePtr



### IsBadStringPtr (3.1)

```
BOOL IsBadStringPtr(lpsz, cchMax)
const void FAR* lpsz;          /* pointer to test */
UINT cchMax;                  /* maximum size of string */
string
/
```

The IsBadStringPtr function determines whether a pointer to a string is valid.

Parameter	Description
lpsz	Points to a null-terminated string.
cchMax	Specifies the maximum size of the string, in bytes.

#### Returns

The return value is nonzero if the pointer is bad (that is, if it does not point to a string of the specified size). The return value is zero if the pointer is good.

#### See Also

IsBadCodePtr, IsBadHugeReadPtr, IsBadHugeWritePtr, IsBadReadPtr, IsBadWritePtr

### IsBadWritePtr (3.1)

```
BOOL IsBadWritePtr(lp, cb)
void FAR* lp;          /* pointer to test          */
UINT cb;              /* number of allocated bytes */
/
```

The IsBadWritePtr function determines whether a pointer to writable memory is valid.

Parameter	Description
lp	Points to the beginning of a block of allocated memory.
cb	Specifies the number of bytes of memory that were allocated.

#### Returns

The return value is nonzero if the pointer is bad (that is, if it does not point to writable memory of the specified size). The return value is zero if the pointer is good.

#### See Also

IsBadCodePtr, IsBadHugeReadPtr, IsBadHugeWritePtr, IsBadReadPtr, IsBadStringPtr

## IsDBCSLeadByte (3.1)

```
BOOL IsDBCSLeadByte(bTestChar)
BYTE bTestChar;    /* character to test */
```

The IsDBCSLeadByte function determines whether a character is a lead byte, the first byte of a character in a double-byte character set (DBCS).

Parameter	Description
bTestChar	Specifies the character to be tested.

### Returns

The return value is nonzero if the character is a DBCS lead byte. Otherwise, it is zero.

### Comments

The language driver for the current language (the language the user selected at setup or by using Control Panel) determines whether the character is in the set. If no language driver is selected, Windows uses an internal function.

Each double-byte character set has a unique set of lead-byte values. By itself, a lead byte has no character value; together, the lead byte and the following byte represent a single character. The second, or following, byte is called a trailing byte.

### See Also

[GetKeyboardType](#)

### IsTask (3.1)

```
BOOL IsTask(htask)
HTASK htask;      /* handle of task */
```

The IsTask function determines whether the given task handle is valid.

Parameter	Description
htask	Identifies a task.

#### Returns

The return value is nonzero if the task handle is valid. Otherwise, it is zero.

## LimitEmsPages (2.x)

```
void LimitEmsPages(cAppKB)
DWORD cAppKB; /* amount of expanded memory available to application */
```

In Windows version 3.1, this function is obsolete and does nothing.

## LoadLibrary (2.x)

HINSTANCE LoadLibrary(lpszLibFileName)  
LPCSTR lpszLibFileName; /\* address of name of library file \*/

The LoadLibrary function loads the specified library module.

Parameter	Description
lpszLibFileName	Points to a null-terminated string that names the library file to be loaded. If the string does not contain a path, Windows searches for the library in this order:

- 1 The current directory.
- 2 The Windows directory (the directory containing WIN.COM); the GetWindowsDirectory function retrieves the path of this directory.
- 3 The Windows system directory (the directory containing such system files as GDI.EXE); the GetSystemDirectory function retrieves the path of this directory.
- 4 The directory containing the executable file for the current task; the GetModuleFileName function retrieves the path of this directory.
- 5 The directories listed in the PATH environment variable.
- 6 The list of directories mapped in a network.

### Returns

The return value is the instance handle of the loaded library module if the function is successful. Otherwise, it is an error value less than HINSTANCE\_ERROR.

### Errors

If the function fails, it returns one of the following error values:

Value	Meaning
0	System was out of memory, executable file was corrupt, or relocations were invalid.
2	File was not found.
3	Path was not found.
5	Attempt was made to dynamically link to a task, or there was a sharing or network-protection error.
6	Library required separate data segments for each task.
8	There was insufficient memory to start the application.
10	Windows version was incorrect.
11	Executable file was invalid. Either it was not a Windows application or there was an error in the .EXE image.
12	Application was designed for a different operating system.
13	Application was designed for MS-DOS 4.0.
14	Type of executable file was unknown.
15	Attempt was made to load a real-mode application (developed for an earlier version of Windows).
16	Attempt was made to load a second instance of an executable file containing multiple data segments that were not marked read-only.
19	Attempt was made to load a compressed executable file. The file must be decompressed before it can be loaded.
20	Dynamic-link library (DLL) file was invalid. One of the DLLs required to run this application was corrupt.
21	Application requires Microsoft Windows 32-bit extensions.

### Comments

If the module has been loaded, LoadLibrary increments (increases by one) the module's reference count. If the module has not been loaded, the function loads it from the specified file.

LoadLibrary increments the reference count for a library module each time an application calls the function. When it has finished using the module, the application should use the FreeLibrary function to decrement (decrease by one) the reference count.

An application can use the GetProcAddress function to access functions in a library that was loaded using LoadLibrary.

#### Example

The following example uses the LoadLibrary function to load the Tool Helper Library TOOLHELP.DLL and the FreeLibrary function to free it:

```
HINSTANCE hinstToolHelp = LoadLibrary("TOOLHELP.DLL");  
  
if ((UINT) hinstToolHelp > 32) {  
    . /* use GetProcAddress to use TOOLHELP functions */  
    .  
}  
else {  
    ErrorHandler();  
}  
  
if ((UINT) hinstToolHelp > 32)  
    FreeLibrary(hinstToolHelp); /* free TOOLHELP.DLL */
```

See Also

FreeLibrary, GetProcAddress

## LoadModule (3.0)

```
HINSTANCE LoadModule(lpszModuleName, lpvParameterBlock)
LPCSTR lpszModuleName; /* address of filename to load */
LPVOID lpvParameterBlock;
/* address of parameter block for new module */
/
```

The LoadModule function loads and executes a Windows application or creates a new instance of an existing Windows application.

Parameter	Description
lpszModuleName	Points to a null-terminated string that contains the complete filename (including the file extension) of the application to be run. If the string does not contain a path, Windows searches for the executable file in this order:

- 1 The current directory.
- 2 The Windows directory (the directory containing WIN.COM), whose path the GetWindowsDirectory function retrieves.
- 3 The Windows system directory (the directory containing such system files as GDI.EXE), whose path the GetSystemDirectory function retrieves.
- 4 The directory containing the executable file for the current task; the GetModuleFileName function obtains the path of this directory.
- 5 The directories listed in the PATH environment variable.
- 6 The list of directories mapped in a network.

lpvParameterBlock	Points to an application-defined LOADPARMS structure that defines the new application's parameter block. The LOADPARMS structure has the following form:
-------------------	--

```
struct _LOADPARMS {
    WORD segEnv; /* child environment */
    LPSTR lpszCmdLine; /* child command tail */
    UINT FAR* lpShow; /* how to show child */
    UINT FAR* lpReserved; /* must be NULL */
} LOADPARMS;
```

Member	Description
segEnv	Specifies whether the child application receives a copy of the parent application's environment or a new environment created by the parent application. If this member is zero, the child application receives an exact duplicate of the parent application's environment block. If the member is nonzero, the value entered must be the segment address of a memory object containing a copy of the new environment for the child application.
lpszCommandLine	Points to a null-terminated string that specifies the command line (excluding the child application name). This string must not exceed 120 characters. If there is no command line, this member must point to a zero-length string (it cannot be set to NULL).
lpShow	Points to an array containing two 16-bit values. The first value must always be set to two. The second value specifies how the application window is to be shown. For a list of the acceptable values, see the description of the nCmdShow parameter of the <u>ShowWindow</u> function.
lpReserved	Reserved; must be NULL.

Returns



The return value is the instance handle of the loaded module if the function is successful. If the function fails, it returns an error value less than `HINSTANCE_ERROR`.

#### Errors

If the function fails, it returns one of the following error values:

Value	Meaning
0	System was out of memory, executable file was corrupt, or relocations were invalid.
2	File was not found.
3	Path was not found.
5	Attempt was made to dynamically link to a task, or there was a sharing or network-protection error.
6	Library required separate data segments for each task.
8	There was insufficient memory to start the application.
10	Windows version was incorrect.
11	Executable file was invalid. Either it was not a Windows application or there was an error in the .EXE image.
12	Application was designed for a different operating system.
13	Application was designed for MS-DOS 4.0.
14	Type of executable file was unknown.
15	Attempt was made to load a real-mode application (developed for an earlier version of Windows).
16	Attempt was made to load a second instance of an executable file containing multiple data segments that were not marked read-only.
19	Attempt was made to load a compressed executable file. The file must be decompressed before it can be loaded.
20	Dynamic-link library (DLL) file was invalid. One of the DLLs required to run this application was corrupt.
21	Application requires Microsoft Windows 32-bit extensions.

#### Comments

The `WinExec` function provides an alternative method for executing an application.

#### Example

The following example uses the `LoadModule` function to run an executable file named `DRAW.EXE`:

```
struct LOADPARMS {
    WORD segEnv; /* child environment */
    LPSTR lpszCmdLine; /* child command tail */
    LPWORD lpwShow; /* how to show child */
    LPWORD lpwReserved; /* must be NULL */
};

char szMsg[80];
HINSTANCE hinstMod;
struct LOADPARMS parms;
WORD awShow[2] = { 2, SW_SHOWMINIMIZED };

parms.segEnv = 0; /* child inherits environment */
parms.lpszCmdLine = (LPSTR) ""; /* no command line */
parms.lpwShow = (LPWORD) awShow; /* shows child as an icon */
parms.lpwReserved = (LPWORD) NULL; /* must be NULL */

hinstMod = LoadModule("draw.exe", &parms);

if ((UINT) hinstMod < 32) {
    sprintf(szMsg, "LoadModule failed; error code = %d",
        hinstMod);
    MessageBox(hwnd, szMsg, "Error", MB_ICONSTOP);
}
else {
    sprintf(szMsg, "LoadModule returned %d", hinstMod);
}
```

```
    MessageBox(hwnd, szMsg, "", MB_OK);  
}
```

See Also

[FreeModule](#), [GetModuleFileName](#), [GetSystemDirectory](#), [GetWindowsDirectory](#), [ShowWindow](#), [WinExec](#)

## LoadResource (2.x)

HGLOBAL LoadResource(hinst, hrsrc)

HINSTANCE hinst; /\* handle of file containing resource \*/

HRSRC hrsrc; /\* handle of

resource

/

\*

The LoadResource function loads the specified resource in global memory.

Parameter	Description
hinst	Identifies an instance of the module whose executable file contains the resource to be loaded.
hrsrc	Identifies the resource to be loaded. This handle must have been created by using the <u>FindResource</u> function.

### Returns

The return value is the instance handle of the global memory object containing the data associated with the resource. It is NULL if no such resource exists.

### Comments

When finished with a resource, an application should free the global memory associated with it by using the FreeResource function.

If the specified resource has been loaded, this function simply increments the reference count for the resource.

The resource is not loaded until the LockResource function is called to translate the handle returned by LoadResource into a far pointer to the resource data.

### See Also

FindResource, FreeResource, LockResource

## LocalAlloc (2.x)

```
HLOCAL LocalAlloc(fuAllocFlags, fuAlloc)
UINT fuAllocFlags;          /* allocation attributes */
UINT fuAlloc;               /* number of bytes to
allocate                    */
/
```

The LocalAlloc function allocates the specified number of bytes from the local heap.

Parameter	Description																						
fuAllocFlags	Specifies how to allocate memory. This parameter can be a combination of the following values:																						
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>LHND</u></td><td>Combines the <u>LMEM_MOVEABLE</u> and <u>LMEM_ZEROINIT</u> flags.</td></tr><tr><td><u>LMEM_DISCARDABLE</u></td><td>Allocates discardable memory.</td></tr><tr><td><u>LMEM_FIXED</u></td><td>Allocates fixed memory. The <u>LMEM_FIXED</u> and <u>LMEM_MOVEABLE</u> flags cannot be combined.</td></tr><tr><td><u>LMEM_MOVEABLE</u></td><td>Allocates movable memory. The <u>LMEM_FIXED</u> and <u>LMEM_MOVEABLE</u> flags cannot be combined.</td></tr><tr><td><u>LMEM_NOCOMPACT</u></td><td>Does not compact or discard memory to satisfy the allocation request.</td></tr><tr><td><u>LMEM_NODISCARD</u></td><td>Does not discard memory to satisfy the allocation request.</td></tr><tr><td><u>LMEM_ZEROINIT</u></td><td>Initializes memory contents to zero.</td></tr><tr><td><u>LPTR</u></td><td>Combines the <u>LMEM_FIXED</u> and <u>LMEM_ZEROINIT</u> flags.</td></tr><tr><td><u>NONZEROLHND</u></td><td>Same as the <u>LMEM_MOVEABLE</u> flag.</td></tr><tr><td><u>NONZEROLPTR</u></td><td>Same as the <u>LMEM_FIXED</u> flag.</td></tr></tbody></table>	Value	Meaning	<u>LHND</u>	Combines the <u>LMEM_MOVEABLE</u> and <u>LMEM_ZEROINIT</u> flags.	<u>LMEM_DISCARDABLE</u>	Allocates discardable memory.	<u>LMEM_FIXED</u>	Allocates fixed memory. The <u>LMEM_FIXED</u> and <u>LMEM_MOVEABLE</u> flags cannot be combined.	<u>LMEM_MOVEABLE</u>	Allocates movable memory. The <u>LMEM_FIXED</u> and <u>LMEM_MOVEABLE</u> flags cannot be combined.	<u>LMEM_NOCOMPACT</u>	Does not compact or discard memory to satisfy the allocation request.	<u>LMEM_NODISCARD</u>	Does not discard memory to satisfy the allocation request.	<u>LMEM_ZEROINIT</u>	Initializes memory contents to zero.	<u>LPTR</u>	Combines the <u>LMEM_FIXED</u> and <u>LMEM_ZEROINIT</u> flags.	<u>NONZEROLHND</u>	Same as the <u>LMEM_MOVEABLE</u> flag.	<u>NONZEROLPTR</u>	Same as the <u>LMEM_FIXED</u> flag.
Value	Meaning																						
<u>LHND</u>	Combines the <u>LMEM_MOVEABLE</u> and <u>LMEM_ZEROINIT</u> flags.																						
<u>LMEM_DISCARDABLE</u>	Allocates discardable memory.																						
<u>LMEM_FIXED</u>	Allocates fixed memory. The <u>LMEM_FIXED</u> and <u>LMEM_MOVEABLE</u> flags cannot be combined.																						
<u>LMEM_MOVEABLE</u>	Allocates movable memory. The <u>LMEM_FIXED</u> and <u>LMEM_MOVEABLE</u> flags cannot be combined.																						
<u>LMEM_NOCOMPACT</u>	Does not compact or discard memory to satisfy the allocation request.																						
<u>LMEM_NODISCARD</u>	Does not discard memory to satisfy the allocation request.																						
<u>LMEM_ZEROINIT</u>	Initializes memory contents to zero.																						
<u>LPTR</u>	Combines the <u>LMEM_FIXED</u> and <u>LMEM_ZEROINIT</u> flags.																						
<u>NONZEROLHND</u>	Same as the <u>LMEM_MOVEABLE</u> flag.																						
<u>NONZEROLPTR</u>	Same as the <u>LMEM_FIXED</u> flag.																						
fuAlloc	Specifies the number of bytes to be allocated.																						

### Returns

The return value is the instance handle of the newly allocated local memory object, if the function is successful. Otherwise, it is NULL.

### Comments

If LocalAlloc allocates movable memory, the return value is a local handle of the memory. To access the memory, an application must use the LocalLock function to convert the handle to a pointer.

If LocalAlloc allocates fixed memory, the return value is a pointer to the memory. To access the memory, an application can simply cast the return value to a pointer.

Fixed memory will be slightly faster than movable memory. If memory will be allocated and freed without an intervening local allocation or reallocation, then the memory should be allocated as fixed.

If this function is successful, it allocates at least the amount requested. If the amount allocated is greater than the amount requested, the application can use the entire amount. To determine the size of a local memory object, an application can use the LocalSize function.

To free a local memory object, an application should use the LocalFree function. To change the size or attributes of an allocated memory object, an application can use the LocalReAlloc function.

### See Also

LocalFree, LocalLock, LocalReAlloc, LocalSize, LocalUnlock

LHND (LMEM\_MOVEABLE | LMEM\_ZEROINIT)

Combines the LMEM\_MOVEABLE and LMEM\_ZEROINIT flags.

LHND (LMEM\_MOVEABLE | LMEM\_ZEROINIT)

LMEM\_DISCARDABLE 0x0F00  
Allocates discardable memory.

LMEM\_DISCARDABLE 0x0F00



LMEM\_FIXED 0x0000

Allocates fixed memory. The LMEM\_FIXED and LMEM\_MOVEABLE flags cannot be combined.

LMEM\_FIXED 0x0000

LMEM\_MOVEABLE 0x0002

Allocates movable memory. The LMEM\_FIXED and LMEM\_MOVEABLE flags cannot be combined.

LMEM\_MOVEABLE 0x0002

LMEM\_NOCOMPACT 0x0010

Does not compact or discard memory to satisfy the allocation request.

LMEM\_NOCOMPACT 0x0010

LMEM\_NODISCARD 0x0020

Does not discard memory to satisfy the allocation request.

LMEM\_NODISCARD 0x0020



LMEM\_ZEROINIT 0x0040

Initializes memory contents to zero.

LMEM\_ZEROINIT 0x0040

LPTR (LMEM\_FIXED | LMEM\_ZEROINIT)

Combines the LMEM\_FIXED and LMEM\_ZEROINIT flags.

LPTR (LMEM\_FIXED | LMEM\_ZEROINIT)

NONZEROLHND (LMEM\_MOVEABLE)

Same as the LMEM\_MOVEABLE flag.

NONZEROLHND (LMEM\_MOVEABLE)

NONZEROLPTR (LMEM\_FIXED)

Same as the LMEM\_FIXED flag.

NONZEROLPTR (LMEM\_FIXED)



## LocalCompact (2.x)

```
UINT LocalCompact(uMinFree)
UINT uMinFree;    /* amount of memory requested */
```

The LocalCompact function rearranges the local heap so that the specified amount of memory is free.

Parameter	Description
uMinFree	Specifies the number of contiguous free bytes requested. If this parameter is zero, the function does not compact memory, but the return value is valid.

### Returns

The return value specifies the number of bytes in the largest free local memory object. If the uMinFree parameter is zero, the return value specifies the number of bytes in the largest free object that Windows can generate if it removes all discardable objects.

### Comments

The function first checks the local heap for the specified number of contiguous free bytes. If the bytes do not exist, the function compacts local memory by moving all unlocked, movable objects into high memory. If this does not generate the requested amount of space, the function discards movable and discardable objects that are not locked, until the requested amount of space is generated (if possible).

### See Also

LocalAlloc, LocalLock

## LocalFlags (2.x)

```
UINT LocalFlags(hloc)  
HLOCAL hloc; /* handle of local memory object */
```

The LocalFlags function retrieves information about the given local memory object.

Parameter	Description
hloc	Identifies the local memory object.

### Returns

The low-order byte of the return value contains the lock count of the object; the high-order byte contains either LMEM\_DISCARDABLE (object has been marked as discardable) or LMEM\_DISCARDED (object has been discarded).

### Comments

To retrieve the lock count from the return value, use the LMEM\_LOCKCOUNT mask.

### See Also

LocalAlloc, LocalLock, LocalReAlloc, LocalUnlock

## LocalFree (2.x)

```
HLOCAL LocalFree(hloc)
HLOCAL hloc;      /* handle of local memory object */
```

The LocalFree function frees the given local memory object (if the object is not locked) and invalidates its handle.

Parameter	Description
hloc	Identifies the local memory object to be freed.

### Returns

The return value is NULL if the function is successful. Otherwise, it is equal to the hloc parameter.

### Comments

An application cannot use the LocalFree function to free a locked memory object--that is, a memory object with a lock count greater than zero.

After freeing the handle of the memory object, an application cannot use the handle again. An attempt to free the same memory object more than once can cause Windows to terminate abnormally.

### See Also

[LocalFlags](#), [LocalLock](#)

## LocalHandle (2.x)

HLOCAL LocalHandle(pvMem)

void NEAR\* pvMem; /\* address of local memory object \*/

The LocalHandle function retrieves the handle of the specified local memory object.

Parameter	Description
-----------	-------------

pvMem	Specifies the address of the local memory object.
-------	---

Returns

The return value is the handle of the specified local memory object if the function is successful. It is NULL if the specified address has no handle.

See Also

[LocalAlloc](#)

■ **LocalInit (2.x)**

```
BOOL LocalInit(uSegment, uStartAddr, uEndAddr)
UINT uSegment;          /* segment to contain local heap */
UINT uStartAddr;       /* starting
address for heap
/
UINT uEndAddr;
* ending address for heap
/
```

The LocalInit function initializes a local heap in the specified segment.

Parameter	Description
uSegment	Identifies the segment that is to contain the local heap.
uStartAddr	Specifies the starting address of the local heap within the segment.
uEndAddr	Specifies the ending address of the local heap within the segment.

**Returns**

The return value is nonzero if the function is successful. Otherwise, it is zero.

**Comments**

The first 16 bytes of the segment containing a local heap must be reserved for use by the system.

**See Also**

GlobalLock, LocalAlloc, LocalReAlloc

Windows 3.1 changes

Removed the discussion of movable data segments (LockData etc.), since this is irrelevant in 3.1. Data segments may or may not move, but Windows maintains their selectors.

## LocalLock (2.x)

```
void NEAR* LocalLock(hloc)
HLOCAL hloc;      /* handle of local memory object */
```

The LocalLock function retrieves a pointer to the given local memory object. LocalLock increments (increases by one) the lock count of movable objects and locks the memory.

Parameter	Description
hloc	Identifies the local memory object to be locked.

### Returns

The return value points to the first byte of memory in the local object, if the function is successful. It is NULL if the object has been discarded or an error occurs.

### Comments

Each time an application calls LocalLock for an object, it must eventually call LocalUnlock for the object.

This function will return NULL if an application attempts to lock a memory object with a size of 0 bytes.

The LocalUnlock function decrements (decreases by one) the lock count for the object if LocalLock incremented the count. Other functions can also affect the lock count of a memory object.

Locked memory will not be moved or discarded unless the memory object is reallocated by the LocalReAlloc function. The object remains locked in memory until its lock count is decreased to zero.

Discarded objects always have a lock count of zero.

### See Also

LocalFlags, LocalReAlloc, LocalUnlock

## LocalReAlloc (2.x)

```
HLOCAL LocalReAlloc(hloc, fuNewSize, fuFlags)
HLOCAL hloc;          /* handle of local memory object */
UINT fuNewSize;      /* new size of
object */
/
UINT fuFlags;
* new allocation attributes
/
```

The LocalReAlloc function changes the size or attributes of the given local memory object.

Parameter	Description												
hloc	Identifies the local memory object to be reallocated.												
fuNewSize	Specifies the new size of the local memory object.												
fuFlags	Specifies how to reallocate the local memory object. If this parameter includes the <code>LMEM_MODIFY</code> and <code>LMEM_DISCARDABLE</code> flags, LocalReAlloc ignores the fuNewSize parameter. The fuFlags parameter can be a combination of the following values.												
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><code>LMEM_DISCARDABLE</code></td><td>Causes a previously movable object to become discardable. This flag can be used only with <code>LMEM_MODIFY</code>.</td></tr><tr><td><code>LMEM_MODIFY</code></td><td>Modifies the object's memory flags. This flag can be used only with <code>LMEM_DISCARDABLE</code>.</td></tr><tr><td><code>LMEM_MOVEABLE</code></td><td>If fuNewSize is zero, this flag causes a previously fixed object to be freed or a previously movable object to be discarded (if the object's lock count is zero). This flag cannot be used with <code>LMEM_MODIFY</code>. If fuNewSize is nonzero and the object identified by the hloc parameter is fixed, this flag allows the reallocated object to be moved to a new fixed location.</td></tr><tr><td><code>LMEM_NOCOMPACT</code></td><td>Prevents memory from being compacted or discarded to satisfy the allocation request. This flag cannot be used with <code>LMEM_MODIFY</code>.</td></tr><tr><td><code>LMEM_ZEROINIT</code></td><td>If the object is growing, this flag causes the additional memory contents to be initialized to zero. This flag cannot be used with <code>LMEM_MODIFY</code>.</td></tr></tbody></table>	Value	Meaning	<code>LMEM_DISCARDABLE</code>	Causes a previously movable object to become discardable. This flag can be used only with <code>LMEM_MODIFY</code> .	<code>LMEM_MODIFY</code>	Modifies the object's memory flags. This flag can be used only with <code>LMEM_DISCARDABLE</code> .	<code>LMEM_MOVEABLE</code>	If fuNewSize is zero, this flag causes a previously fixed object to be freed or a previously movable object to be discarded (if the object's lock count is zero). This flag cannot be used with <code>LMEM_MODIFY</code> . If fuNewSize is nonzero and the object identified by the hloc parameter is fixed, this flag allows the reallocated object to be moved to a new fixed location.	<code>LMEM_NOCOMPACT</code>	Prevents memory from being compacted or discarded to satisfy the allocation request. This flag cannot be used with <code>LMEM_MODIFY</code> .	<code>LMEM_ZEROINIT</code>	If the object is growing, this flag causes the additional memory contents to be initialized to zero. This flag cannot be used with <code>LMEM_MODIFY</code> .
Value	Meaning												
<code>LMEM_DISCARDABLE</code>	Causes a previously movable object to become discardable. This flag can be used only with <code>LMEM_MODIFY</code> .												
<code>LMEM_MODIFY</code>	Modifies the object's memory flags. This flag can be used only with <code>LMEM_DISCARDABLE</code> .												
<code>LMEM_MOVEABLE</code>	If fuNewSize is zero, this flag causes a previously fixed object to be freed or a previously movable object to be discarded (if the object's lock count is zero). This flag cannot be used with <code>LMEM_MODIFY</code> . If fuNewSize is nonzero and the object identified by the hloc parameter is fixed, this flag allows the reallocated object to be moved to a new fixed location.												
<code>LMEM_NOCOMPACT</code>	Prevents memory from being compacted or discarded to satisfy the allocation request. This flag cannot be used with <code>LMEM_MODIFY</code> .												
<code>LMEM_ZEROINIT</code>	If the object is growing, this flag causes the additional memory contents to be initialized to zero. This flag cannot be used with <code>LMEM_MODIFY</code> .												

### Returns

The return value is the handle of the reallocated local memory object, if the function is successful. Otherwise, it is NULL.

### Comments

If LocalReAlloc reallocates a movable object, the return value is a local handle of the memory. To access the memory, an application must use the [LocalLock](#) function to convert the handle to a pointer.

If LocalReAlloc reallocates a fixed object, the return value is a pointer to the memory. To access the memory, an application can simply cast the return value to a pointer.

To free a local memory object, an application should use the [LocalFree](#) function.

### See Also

[LocalAlloc](#), [LocalDiscard](#), [LocalFree](#), [LocalLock](#)



## LocalShrink (2.x)

```
UINT LocalShrink(hloc, cbNewSize)
HLOCAL hloc;          /* segment containing local heap */
UINT cbNewSize;      /* new size of local heap */
/
```

The LocalShrink function shrinks the local heap in the given segment.

Parameter	Description
hloc	Identifies the segment that contains the local heap. If this parameter is zero, the function shrinks the heap in the current data segment.
cbNewSize	Specifies the new size, in bytes, of the local heap.

### Returns

The return value specifies the new size of the local heap if the function is successful.

### Comments

Windows will not shrink the portion of the data segment that contains the stack and the static variables.

Use the GlobalSize function to determine the new size of the data segment.

### See Also

GlobalSize

## LocalSize (2.x)

```
UINT LocalSize(hloc)
HLOCAL hloc;      /* handle of local memory object */
```

The LocalSize function returns the current size, in bytes, of the given local memory object.

Parameter	Description
hloc	Identifies the local memory object.

### Returns

The return value specifies the size, in bytes, of the memory object, if the function is successful. It is zero if the specified handle is invalid or if the object has been discarded.

### Comments

The size of a memory object sometimes is larger than the size requested when the memory was allocated.

To verify that the memory object has not been discarded, an application should call the LocalFlags function prior to calling the LocalSize function. If the memory object has been discarded, the return value for LocalSize is meaningless.

### See Also

LocalAlloc, LocalFlags

## LocalUnlock (2.x)

```
BOOL LocalUnlock(hloc)
HLOCAL hloc;      /* handle of local memory object */
```

The LocalUnlock function unlocks the given local memory object. This function has no effect on fixed memory.

Parameter	Description
hloc	Identifies the local memory object to be unlocked.

### Returns

The return value is zero if the function is successful. Otherwise, it is nonzero.

### Comments

With discardable memory, this function decrements (decreases by one) the object's lock count. The object is completely unlocked, and subject to discarding, if the lock count is decreased to zero.

### See Also

[LocalLock](#)

## LockResource (2.x)

```
void FAR* LockResource(hglb)
HGLOBAL hglb;      /* handle of resource */
```

The LockResource function locks the given resource. The resource is locked in memory and its reference count is incremented (increased by one). The locked resource is not subject to discarding.

Parameter	Description
hglb	Identifies the resource to be locked. This handle must have been created by using the <u>LoadResource</u> function.

### Returns

The return value points to the first byte of the loaded resource if the function is successful. Otherwise, it is NULL.

### Comments

The resource remains locked in memory until its reference count is decreased to zero by calls to the FreeResource function.

If the resource identified by the hglb parameter has been discarded, the resource-handler function (if any) associated with the resource is called before the LockResource function returns. The resource-handler function can recalculate and reload the resource if necessary. After the resource-handler function returns, LockResource makes another attempt to lock the resource and returns with the result.

Using the handle returned by the FindResource function for the hglb parameter causes an error.

Use the UnlockResource macro to unlock a resource that was locked by LockResource.

### See Also

FindResource, FreeResource, SetResourceHandler

## LockSegment (2.x)

```
HGLOBAL LockSegment(uSegment)
UINT uSegment;          /* segment to lock */
```

The LockSegment function locks the specified discardable segment. The segment is locked into memory at the given address and its lock count is incremented (increased by one).

Parameter	Description
uSegment	Specifies the segment address of the segment to be locked. If this parameter is -1, the LockSegment function locks the current data segment.

### Returns

The return value specifies the data segment if the function is successful. It is NULL if the segment has been discarded or an error occurs.

### Comments

Locked memory is not subject to discarding except when a portion of the segment is being reallocated by the GlobalReAlloc function. The segment remains locked in memory until its lock count is decreased to zero by the UnlockSegment function.

Each time an application calls LockSegment for a segment, it must eventually call UnlockSegment for the segment. The UnlockSegment function decrements the lock count for the segment. Other functions also can affect the lock count of a memory object. For a list of these functions, see the description of the GlobalFlags function.

### See Also

GlobalFlags, GlobalReAlloc, LockData, UnlockSegment

### LogError (3.1)

```
void LogError(uErr, lpvInfo)
UINT uErr;          /* error type */
void FAR* lpvInfo; /* address of
error information */
/
```

The LogError function identifies the most recent system error. An application's interrupt callback function typically calls LogError to return error information to the user.

Parameter	Description
uErr	Specifies the type of error that occurred. The lpvInfo parameter may point to more information about the error, depending on the value of uErr. This parameter may be one or more of the following values:

Value	Meaning
ERR_ALLOCRES	<u>AllocResource</u> failed.
ERR_BADINDEX	Bad index to <u>GetClassLong</u> , <u>GetClassWord</u> , <u>GetWindowLong</u> , <u>GetWindowWord</u> , <u>SetClassLong</u> , <u>SetClassWord</u> , <u>SetWindowLong</u> , or <u>SetWindowWord</u> .
ERR_BYTE	Invalid 8-bit parameter.
ERR_CREATEDC	<u>CreateCompatibleDC</u> , <u>CreateDC</u> , or <u>CreateIC</u> failed.
ERR_CREATEDLG	Could not create dialog box because <u>LoadMenu</u> failed.
ERR_CREATEDLG2	Could not create dialog box because <u>CreateWindow</u> failed.
ERR_CREATEMENU	Could not create menu.
ERR_CREATEMETA	<u>CreateMetaFile</u> failed.
ERR_CREATEWND	Could not create window because the class was not found.
ERR_DCBUSY	Device context (DC) cache is full.
ERR_DELOBJSELECTED	Program is trying to delete a bitmap that is selected into the DC.
ERR_DWORD	Invalid 32-bit parameter.
ERR_GALLOC	<u>GlobalAlloc</u> failed.
ERR_GLOCK	<u>GlobalLock</u> failed.
ERR_GREALLOC	<u>GlobalReAlloc</u> failed.
ERR_LALLOC	<u>LocalAlloc</u> failed.
ERR_LLOCK	<u>LocalLock</u> failed.
ERR_LOADMENU	<u>LoadMenu</u> failed.
ERR_LOADMODULE	<u>LoadModule</u> failed.
ERR_LOADSTR	<u>LoadString</u> failed.
ERR_LOCKRES	<u>LockResource</u> failed.
ERR_LREALLOC	<u>LocalReAlloc</u> failed.
ERR_NESTEDBEGINPAINT	Program contains nested <u>BeginPaint</u> calls.
ERR_REGISTERCLASS	<u>RegisterClass</u> failed because the class is already registered.
ERR_SELBITMAP	Program is trying to select a bitmap that is already selected.
ERR_SIZE_MASK	Identifies which 2 bits of uErr specify the size of the invalid parameter.
ERR_STRUCEXTRA	Program is using unallocated space.
ERR_WARNING	A non-fatal error occurred.
ERR_WORD	Invalid 16-bit parameter.

**lpvInfo** Points to more information about the error. The value of lpvInfo depends on the value of uErr. If the value of (uErr & ERR\_SIZE\_MASK) is 0, lpvInfo is undefined. Currently, no uErr code has defined meanings for lpvInfo.

**Returns**

This function does not return a value.

**Comments**

The errors identified by LogError may be trapped by the callback function that NotifyRegister installs.

Error values whose low 12 bits are less than 0x07FF are reserved for use by Windows.

**See Also**

LogParamError, NotifyRegister

## LogParamError (3.1)

```
void LogParamError(uErr, lpfm, lpvParam)
UINT uErr; /* error type */
FARPROC lpfm; /* address where error occurred */
void FAR* lpvParam; /* address of more error information */
```

The LogParamError function identifies the most recent parameter validation error. An application's interrupt callback function typically calls LogParamError to return information about an invalid parameter to the user.

Parameter	Description																																																																						
uErr	Specifies the type of parameter validation error that occurred. The lpvParam parameter may point to more information about the error, depending on the value of uErr. This parameter may be one or more of the following values:																																																																						
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>ERR_BAD_ATOM</u></td><td>Invalid atom.</td></tr><tr><td><u>ERR_BAD_CID</u></td><td>Invalid communications identifier (CID).</td></tr><tr><td><u>ERR_BAD_COORDS</u></td><td>Invalid x,y coordinates.</td></tr><tr><td><u>ERR_BAD_DEFLAGS</u></td><td>Invalid 32-bit flags.</td></tr><tr><td><u>ERR_BAD_DINDEX</u></td><td>Invalid 32-bit index or index out-of-range.</td></tr><tr><td><u>ERR_BAD_DVALUE</u></td><td>Invalid 32-bit signed or unsigned value.</td></tr><tr><td><u>ERR_BAD_FLAGS</u></td><td>Invalid bit flags.</td></tr><tr><td><u>ERR_BAD_FUNC_PTR</u></td><td>Invalid function pointer.</td></tr><tr><td><u>ERR_BAD_GDI_OBJECT</u></td><td>Invalid graphics device interface (GDI) object.</td></tr><tr><td><u>ERR_BAD_GLOBAL_HANDLE</u></td><td>Invalid global handle.</td></tr><tr><td><u>ERR_BAD_HANDLE</u></td><td>Invalid generic handle.</td></tr><tr><td><u>ERR_BAD_HBITMAP</u></td><td>Invalid bitmap handle.</td></tr><tr><td><u>ERR_BAD_HBRUSH</u></td><td>Invalid brush handle.</td></tr><tr><td><u>ERR_BAD_HCURSOR</u></td><td>Invalid cursor handle.</td></tr><tr><td><u>ERR_BAD_HDC</u></td><td>Invalid device context (DC) handle.</td></tr><tr><td><u>ERR_BAD_HDRVR</u></td><td>Invalid driver handle.</td></tr><tr><td><u>ERR_BAD_HDWP</u></td><td>Invalid handle of a window-position structure.</td></tr><tr><td><u>ERR_BAD_HFILE</u></td><td>Invalid file handle.</td></tr><tr><td><u>ERR_BAD_HFONT</u></td><td>Invalid font handle.</td></tr><tr><td><u>ERR_BAD_HICON</u></td><td>Invalid icon handle.</td></tr><tr><td><u>ERR_BAD_HINSTANCE</u></td><td>Invalid instance handle.</td></tr><tr><td><u>ERR_BAD_HMENU</u></td><td>Invalid menu handle.</td></tr><tr><td><u>ERR_BAD_HMETAFILE</u></td><td>Invalid metafile handle.</td></tr><tr><td><u>ERR_BAD_HMODULE</u></td><td>Invalid module handle.</td></tr><tr><td><u>ERR_BAD_HPALETTE</u></td><td>Invalid palette handle.</td></tr><tr><td><u>ERR_BAD_HPEN</u></td><td>Invalid pen handle.</td></tr><tr><td><u>ERR_BAD_HRGN</u></td><td>Invalid region handle.</td></tr><tr><td><u>ERR_BAD_HWND</u></td><td>Invalid window handle.</td></tr><tr><td><u>ERR_BAD_INDEX</u></td><td>Invalid index or index out-of-range.</td></tr><tr><td><u>ERR_BAD_LOCAL_HANDLE</u></td><td>Invalid local handle.</td></tr><tr><td><u>ERR_BAD_PTR</u></td><td>Invalid pointer.</td></tr><tr><td><u>ERR_BAD_SELECTOR</u></td><td>Invalid selector.</td></tr><tr><td><u>ERR_BAD_STRING_PTR</u></td><td>Invalid zero-terminated string pointer.</td></tr><tr><td><u>ERR_BAD_VALUE</u></td><td>Invalid 16-bit signed or unsigned value.</td></tr></tbody></table>	Value	Meaning	<u>ERR_BAD_ATOM</u>	Invalid atom.	<u>ERR_BAD_CID</u>	Invalid communications identifier (CID).	<u>ERR_BAD_COORDS</u>	Invalid x,y coordinates.	<u>ERR_BAD_DEFLAGS</u>	Invalid 32-bit flags.	<u>ERR_BAD_DINDEX</u>	Invalid 32-bit index or index out-of-range.	<u>ERR_BAD_DVALUE</u>	Invalid 32-bit signed or unsigned value.	<u>ERR_BAD_FLAGS</u>	Invalid bit flags.	<u>ERR_BAD_FUNC_PTR</u>	Invalid function pointer.	<u>ERR_BAD_GDI_OBJECT</u>	Invalid graphics device interface (GDI) object.	<u>ERR_BAD_GLOBAL_HANDLE</u>	Invalid global handle.	<u>ERR_BAD_HANDLE</u>	Invalid generic handle.	<u>ERR_BAD_HBITMAP</u>	Invalid bitmap handle.	<u>ERR_BAD_HBRUSH</u>	Invalid brush handle.	<u>ERR_BAD_HCURSOR</u>	Invalid cursor handle.	<u>ERR_BAD_HDC</u>	Invalid device context (DC) handle.	<u>ERR_BAD_HDRVR</u>	Invalid driver handle.	<u>ERR_BAD_HDWP</u>	Invalid handle of a window-position structure.	<u>ERR_BAD_HFILE</u>	Invalid file handle.	<u>ERR_BAD_HFONT</u>	Invalid font handle.	<u>ERR_BAD_HICON</u>	Invalid icon handle.	<u>ERR_BAD_HINSTANCE</u>	Invalid instance handle.	<u>ERR_BAD_HMENU</u>	Invalid menu handle.	<u>ERR_BAD_HMETAFILE</u>	Invalid metafile handle.	<u>ERR_BAD_HMODULE</u>	Invalid module handle.	<u>ERR_BAD_HPALETTE</u>	Invalid palette handle.	<u>ERR_BAD_HPEN</u>	Invalid pen handle.	<u>ERR_BAD_HRGN</u>	Invalid region handle.	<u>ERR_BAD_HWND</u>	Invalid window handle.	<u>ERR_BAD_INDEX</u>	Invalid index or index out-of-range.	<u>ERR_BAD_LOCAL_HANDLE</u>	Invalid local handle.	<u>ERR_BAD_PTR</u>	Invalid pointer.	<u>ERR_BAD_SELECTOR</u>	Invalid selector.	<u>ERR_BAD_STRING_PTR</u>	Invalid zero-terminated string pointer.	<u>ERR_BAD_VALUE</u>	Invalid 16-bit signed or unsigned value.
Value	Meaning																																																																						
<u>ERR_BAD_ATOM</u>	Invalid atom.																																																																						
<u>ERR_BAD_CID</u>	Invalid communications identifier (CID).																																																																						
<u>ERR_BAD_COORDS</u>	Invalid x,y coordinates.																																																																						
<u>ERR_BAD_DEFLAGS</u>	Invalid 32-bit flags.																																																																						
<u>ERR_BAD_DINDEX</u>	Invalid 32-bit index or index out-of-range.																																																																						
<u>ERR_BAD_DVALUE</u>	Invalid 32-bit signed or unsigned value.																																																																						
<u>ERR_BAD_FLAGS</u>	Invalid bit flags.																																																																						
<u>ERR_BAD_FUNC_PTR</u>	Invalid function pointer.																																																																						
<u>ERR_BAD_GDI_OBJECT</u>	Invalid graphics device interface (GDI) object.																																																																						
<u>ERR_BAD_GLOBAL_HANDLE</u>	Invalid global handle.																																																																						
<u>ERR_BAD_HANDLE</u>	Invalid generic handle.																																																																						
<u>ERR_BAD_HBITMAP</u>	Invalid bitmap handle.																																																																						
<u>ERR_BAD_HBRUSH</u>	Invalid brush handle.																																																																						
<u>ERR_BAD_HCURSOR</u>	Invalid cursor handle.																																																																						
<u>ERR_BAD_HDC</u>	Invalid device context (DC) handle.																																																																						
<u>ERR_BAD_HDRVR</u>	Invalid driver handle.																																																																						
<u>ERR_BAD_HDWP</u>	Invalid handle of a window-position structure.																																																																						
<u>ERR_BAD_HFILE</u>	Invalid file handle.																																																																						
<u>ERR_BAD_HFONT</u>	Invalid font handle.																																																																						
<u>ERR_BAD_HICON</u>	Invalid icon handle.																																																																						
<u>ERR_BAD_HINSTANCE</u>	Invalid instance handle.																																																																						
<u>ERR_BAD_HMENU</u>	Invalid menu handle.																																																																						
<u>ERR_BAD_HMETAFILE</u>	Invalid metafile handle.																																																																						
<u>ERR_BAD_HMODULE</u>	Invalid module handle.																																																																						
<u>ERR_BAD_HPALETTE</u>	Invalid palette handle.																																																																						
<u>ERR_BAD_HPEN</u>	Invalid pen handle.																																																																						
<u>ERR_BAD_HRGN</u>	Invalid region handle.																																																																						
<u>ERR_BAD_HWND</u>	Invalid window handle.																																																																						
<u>ERR_BAD_INDEX</u>	Invalid index or index out-of-range.																																																																						
<u>ERR_BAD_LOCAL_HANDLE</u>	Invalid local handle.																																																																						
<u>ERR_BAD_PTR</u>	Invalid pointer.																																																																						
<u>ERR_BAD_SELECTOR</u>	Invalid selector.																																																																						
<u>ERR_BAD_STRING_PTR</u>	Invalid zero-terminated string pointer.																																																																						
<u>ERR_BAD_VALUE</u>	Invalid 16-bit signed or unsigned value.																																																																						



<u>ERR_BYTE</u>	Invalid 8-bit parameter.
<u>ERR_DWORD</u>	Invalid 32-bit parameter.
<u>ERR_PARAM</u>	A parameter validation error occurred. This flag is always set.
<u>ERR_SIZE_MASK</u>	Identifies which 2 bits of uErr specify the size of the invalid parameter.
<u>ERR_WARNING</u>	An invalid parameter was detected, but the error is not serious enough to cause the function to fail. The invalid parameter is reported, but the call runs as usual.
<u>ERR_WORD</u>	Invalid 16-bit parameter.

lpfn Specifies the address at which the parameter error occurred. This value is NULL if the address is unknown.

lpvParam Points to more information about the error. The value of lpvParam depends on the value of uErr. If the value of (uErr & ERR\_SIZE\_MASK) is 0, lpvParam is undefined. Currently, no uErr code has defined meanings for lpvParam.

#### Returns

This function does not return a value.

#### Comments

The errors identified by LogParamError may be trapped by the callback function that NotifyRegister installs.

Error values whose low 12 bits are less than 0x07FF are reserved for use by Windows.

The size of the value passed in lpvParam is determined by the values of the bits selected by ERR\_SIZE\_MASK, as follows:

```
switch (err & ERR_SIZE_MASK)
{
case ERR_BYTE: /* 8-bit invalid parameter */
    b = LOBYTE(param);
    break;

case ERR_WORD: /* 16-bit invalid parameter */
    w = LOWORD(param);
    break;

case ERR_DWORD: /* 32-bit invalid parameter */
    l = (DWORD)param;
    break;

default: /* invalid parameter value is unknown */
    break;
}
```

#### See Also

LogError, NotifyRegister

ERR\_BAD\_ATOM 0x6024

Invalid atom.

ERR\_BAD\_ATOM 0x6024

ERR\_BAD\_CID 0x6045

Invalid communications identifier (CID).

ERR\_BAD\_CID 0x6045

ERR\_BAD\_COORDS 0x7060

Invalid x,y coordinates.

ERR\_BAD\_COORDS 0x7060

ERR\_BAD\_DFLAGS 0x7005

Invalid 32-bit flags.



ERR\_BAD\_DFLAGS 0x7005

ERR\_BAD\_DINDEX 0x7006

Invalid 32-bit index or index out-of-range.

ERR\_BAD\_DINDEX 0x7006

ERR\_BAD\_DVALUE 0x7004

Invalid 32-bit signed or unsigned value.

ERR\_BAD\_DVALUE 0x7004

ERR\_BAD\_FLAGS 0x6002

Invalid bit flags.

ERR\_BAD\_FLAGS 0x6002

ERR\_BAD\_FUNC\_PTR 0x7008

Invalid function pointer.



ERR\_BAD\_FUNC\_PTR 0x7008

ERR\_BAD\_GDI\_OBJECT 0x6061  
Invalid graphics device interface (GDI) object.

ERR\_BAD\_GDI\_OBJECT 0x6061

ERR\_BAD\_GLOBAL\_HANDLE 0x6022

Invalid global handle.

ERR\_BAD\_GLOBAL\_HANDLE 0x6022

ERR\_BAD\_HANDLE 0x600b

Invalid generic handle.

ERR\_BAD\_HANDLE 0x600b

ERR\_BAD\_HBITMAP 0x6066

Invalid bitmap handle.



ERR\_BAD\_HBITMAP 0x6066

ERR\_BAD\_HBRUSH 0x6065

Invalid brush handle.

ERR\_BAD\_HBRUSH 0x6065

ERR\_BAD\_HCURSOR 0x6042

Invalid cursor handle.

ERR\_BAD\_HCURSOR 0x6042

ERR\_BAD\_HDC 0x6062

Invalid device context (DC) handle.

ERR\_BAD\_HDC 0x6062

ERR\_BAD\_HDRV 0x6046

Invalid driver handle.



ERR\_BAD\_HDRV 0x6046

ERR\_BAD\_HDWP 0x6044

Invalid handle of a window-position structure.

ERR\_BAD\_HDWP 0x6044

ERR\_BAD\_HFILE 0x6025

Invalid file handle.

ERR\_BAD\_HFILE 0x6025

ERR\_BAD\_HFONT 0x6064

Invalid font handle.

ERR\_BAD\_HFONT 0x6064

ERR\_BAD\_HICON 0x6043

Invalid icon handle.



ERR\_BAD\_HICON 0x6043

ERR\_BAD\_HINSTANCE 0x6020

Invalid instance handle.

ERR\_BAD\_HINSTANCE 0x6020

ERR\_BAD\_HMENU 0x6041

Invalid menu handle.

ERR\_BAD\_HMENU 0x6041

ERR\_BAD\_HMETAFILE 0x6069

Invalid metafile handle.

ERR\_BAD\_HMETAFILE 0x6069

ERR\_BAD\_HMODULE 0x6021

Invalid module handle.



ERR\_BAD\_HMODULE 0x6021

ERR\_BAD\_HPALETTE 0x6068

Invalid palette handle.

ERR\_BAD\_HPALETTE 0x6068

ERR\_BAD\_HPEN 0x6063

Invalid pen handle.

ERR\_BAD\_HPEN 0x6063

ERR\_BAD\_HRGN 0x6067  
Invalid region handle.

ERR\_BAD\_HRGN 0x6067

ERR\_BAD\_HWND 0x6040

Invalid window handle.



ERR\_BAD\_HWND 0x6040

ERR\_BAD\_INDEX 0x6003

Invalid index or index out-of-range.

ERR\_BAD\_INDEX 0x6003

ERR\_BAD\_LOCAL\_HANDLE 0x6023

Invalid local handle.

ERR\_BAD\_LOCAL\_HANDLE 0x6023

ERR\_BAD\_PTR 0x7007

Invalid pointer.

ERR\_BAD\_PTR 0x7007

ERR\_BAD\_SELECTOR 0x6009

Invalid selector.



ERR\_BAD\_SELECTOR 0x6009

ERR\_BAD\_STRING\_PTR 0x700a  
Invalid zero-terminated string pointer.

ERR\_BAD\_STRING\_PTR 0x700a

ERR\_BAD\_VALUE 0x6001

Invalid 16-bit signed or unsigned value.

ERR\_BAD\_VALUE 0x6001

ERR\_BYTE 0x1000  
Invalid 8-bit parameter.

ERR\_BYTE 0x1000

ERR\_DWORD 0x3000  
Invalid 32-bit parameter.



ERR\_DWORD 0x3000

ERR\_PARAM 0x4000

A parameter validation error occurred. This flag is always set.

ERR\_PARAM 0x4000

ERR\_SIZE\_MASK 0x3000

Identifies which 2 bits of uErr specify the size of the invalid parameter.

ERR\_SIZE\_MASK 0x3000

ERR\_WARNING 0x8000

An invalid parameter was detected, but the error is not serious enough to cause the function to fail. The invalid parameter is reported, but the call runs as usual.

ERR\_WARNING 0x8000

ERR\_WORD 0x2000  
Invalid 16-bit parameter.



ERR\_WORD 0x2000

## lstrcat (2.x)

```
LPSTR lstrcat(lpszString1, lpszString2)
LPSTR lpszString1;          /* address of buffer for concatenated strings */
LPCSTR lpszString2;
/* address of string to add to string1
/
```

The lstrcat function appends one string to another.

Parameter	Description
lpszString1	Points to a byte array containing a null-terminated string. The byte array containing the string must be large enough to contain both strings.
lpszString2	Points to the null-terminated string to be appended to the string specified in the lpszString1 parameter.

### Returns

The return value points to lpszString1 if the function is successful.

### Comments

Both strings must be less than 64K in size.

### Example

The following example uses the lstrcat function to append a test string to a buffer:

```
char szBuf[80] = { "the test string is " };

lstrcat(szBuf, lpsz);
MessageBox(hwnd, szBuf, "lstrcat", MB_OK);
```

### See Also

[lstrcpy](#)

## lstrcpy (2.x)

```
LPSTR lstrcpy(lpszString1, lpszString2)
LPSTR lpszString1;          /* address of buffer */
LPCSTR lpszString2;        /* address of
string to copy */
/
```

The lstrcpy function copies a string to a buffer.

Parameter	Description
lpszString1	Points to a buffer that will receive the contents of the string pointed to by the lpszString2 parameter. The buffer must be large enough to contain the string, including the terminating null character.
lpszString2	Points to the null-terminated string to be copied.

### Returns

The return value is a pointer to lpszString1 if the function is successful. Otherwise, it is NULL.

### Comments

This function can be used to copy a double-byte character set (DBCS) string.

Both strings must be less than 64K in size.

### See Also

[lstrcat](#), [lstrcpyn](#), [lstrlen](#)

### lstrcpy (3.1)

```
LPSTR lstrcpy(lpszString1, lpszString2, cChars)
LPSTR lpszString1; /* address of buffer */
LPCSTR lpszString2; /* address of string to copy from */
int cChars;
/* number of characters to copy */
```

The lstrcpy function copies a specified number of characters in a string to a buffer.

Parameter	Description
lpszString1	Points to a buffer that will receive characters from the string pointed to by the lpszString2 parameter.
lpszString2	Points to the null-terminated string to copy from.
cChars	Specifies the number of characters to copy from the string pointed to by the lpszString2 parameter.

#### Returns

The return value is a pointer to lpszString1 if the function is successful. Otherwise, it is NULL.

#### See Also

[lstrcpy](#)

## lstrlen (2.x)

```
int lstrlen(lpszString)
LPCSTR lpszString;      /* address of string to count */
```

The lstrlen function returns the length, in bytes, of the specified string (not including the terminating null character).

Parameter	Description
lpszString	Points to a null-terminated string. This string must be less than 64K in size.

### Returns

The return value specifies the length, in bytes, of the string pointed to by the lpszString parameter. There is no error return.

### See Also

[lstrcpy](#)

## MakeProcInstance (2.x)

```
FARPROC MakeProcInstance(lpProc, hinst)
FARPROC lpProc;          /* address of function          */
HINSTANCE hinst;         /* instance to bind to function */
/
```

The `MakeProcInstance` function returns the address of the prolog code for an exported function. The prolog code binds an instance data segment to an exported function. When the function is called, it has access to variables and data in that instance data segment.

Parameter	Description
<code>lpProc</code>	Specifies the address of an exported function.
<code>hinst</code>	Identifies the instance associated with the desired data segment.

### Returns

The return value points to the prolog code for the specified exported function, if `MakeProcInstance` is successful. Otherwise, it is `NULL`.

### Comments

The `MakeProcInstance` function is used to retrieve a calling address for a function that must be called by Windows, such as an About procedure. This function must be used only to access functions from instances of the current module. If the address specified in the `lpProc` parameter identifies a procedure in a dynamic-link library, `MakeProcInstance` returns the same address specified in `lpProc`.

After `MakeProcInstance` has been called for a particular function, all calls to that function should be made through the retrieved address.

The `FreeProcInstance` function frees the function from the data segment bound to it by the `MakeProcInstance` function.

`MakeProcInstance` will create more than one procedure instance. To avoid wasting memory, an application should not call `MakeProcInstance` more than once using the same function and instance handle.

If you are using a recent version of a Windows compiler, you may not have to use the `MakeProcInstance` function. Consult your compiler manual for specific information on function prolog and epilog code.

### See Also

[FreeProcInstance](#), [GetProcAddress](#)

## MapVirtualKey (3.0)

```
UINT MapVirtualKey(uKeyCode, fuMapType)
UINT uKeyCode;      /* virtual-key code or scan code */
UINT fuMapType;     /* translation to perform */
```

The MapVirtualKey function translates (maps) a virtual-key code into a scan code or ASCII value, or it translates a scan code into a virtual-key code.

Parameter	Description
uKeyCode	Specifies the virtual-key code or scan code for a key. How this parameter is interpreted depends on the value of the fuMapType parameter.
fuMapType	Specifies the translation to perform. If this parameter is 0, the uKeyCode parameter is a virtual-key code and is translated into its corresponding scan code. If fuMapType is 1, uKeyCode is a scan code and is translated to a virtual-key code. If fuMapType is 2, uKeyCode is a virtual-key code and is translated to an unshifted ASCII value. Other values are reserved.

### Returns

The return value depends on the value of the uKeyCode and fuMapType parameters. For more information, see the description of the fuMapType parameter.

### See Also

[OemKeyScan](#), [VkKeyScan](#)

### MulDiv (3.0)

```
int MulDiv(nMultiplicand, nMultiplier, nDivisor)
int nMultiplicand; /* 16-bit signed multiplicand */
int nMultiplier; /* 16-bit signed multiplier */
int nDivisor; /* 16-bit signed divisor */
```

The MulDiv function multiplies two 16-bit values and then divides the 32-bit result by a third 16-bit value. The return value is the 16-bit result of the division, rounded up or down to the nearest integer.

Parameter	Description
nMultiplicand	Specifies the multiplicand.
nMultiplier	Specifies the multiplier.
nDivisor	Specifies the number by which the result of the multiplication (nMultiplicand * nMultiplier) is to be divided.

#### Returns

The return value is the result of the multiplication and division if the function is successful. The return value is -32,768 if either an overflow occurs or the nDivisor parameter is 0.

#### See Also

[CreateFontIndirect](#), [GetDeviceCaps](#)



## NetBIOSCall (3.0)

The NetBIOSCall function allows an application to issue the NETBIOS Interrupt 5Ch. This function can be called only from assembly-language routines. It is exported from KRNL286.EXE and KRNL386.EXE and is not defined in any Windows header files.

### Parameters

Registers must be set up as required by Interrupt 5Ch before the application calls the NetBIOSCall function.

### Returns

The register contents are preserved as they are returned by Interrupt 5Ch.

### Comments

Applications should use this function instead of directly issuing a NETBIOS Interrupt 5Ch.

### Example

To use this function, an application should declare it in an assembly-language routine, as follows:

```
extrn NETBIOSCALL: far
```

If the application includes CMACROS.INC, the function is declared as follows:

```
externFP NetBIOSCall
```

Following is an example of how to use the NetBIOSCall function:

```
extrn NETBIOSCALL: far
```

```
·
```

```
·
```

```
·
```

```
    ;set registers
```

```
    cCall NetBIOSCall
```

## OemKeyScan (3.0)

```
DWORD OemKeyScan(uOemChar)
UINT uOemChar;      /* OEM ASCII character */
```

The OemKeyScan function translates (maps) OEM ASCII codes 0 through 0xFF to their corresponding OEM scan codes and shift states.

Parameter	Description
uOemChar	Specifies the ASCII value of the OEM character.

### Returns

The low-order word of the return value contains the scan code of the specified OEM character; the high-order word contains flags that indicate the shift state: If bit 1 is set, a SHIFT key is pressed; if bit 2 is set, a CTRL key is pressed. Both the low-order and high-order words of the return value contain -1 if the character is not defined in the OEM character tables.

### Comments

The OemKeyScan function does not translate characters that require CTRL+ALT or dead keys. Characters not translated by this function must be copied by simulating input, using the ALT+ keypad mechanism. For this to work, the NUM LOCK key must be off.

This function calls the VkKeyScan function in recent versions of the keyboard device drivers.

OemKeyScan allows an application to send OEM text to another application by simulating keyboard input. It is used specifically for this purpose by Windows in 386 enhanced mode.

### See Also

VkKeyScan

## OemToAnsi (2.x)

```
void OemToAnsi(hpszOemStr, hpszWindowsStr)
const char _huge* hpszOemStr;          /* address of string to translate */
char _huge* hpszWindowsStr;          /* address of translated string buffer */
/
```

The OemToAnsi function translates a string from the OEM-defined character set into the Windows character set.

Parameter	Description
hpszOemStr	Points to a null-terminated string of characters from the OEM-defined character set.
hpszWindowsStr	Points to the location where the translated string is to be copied. To translate the string in place, the hpszWindowsStr parameter can be the same as the hpszOemStr parameter.

### Returns

This function does not return a value.

### See Also

[AnsiToOem](#), [OemToAnsiBuff](#)

## OemToAnsiBuff (2.x)

```
void OemToAnsiBuff(lpszOemStr, lpszWindowsStr, cbOemStr)
LPCSTR lpszOemStr;          /* address of OEM character string */
LPSTR lpszWindowsStr;      /* address of buffer for Windows string */
/
UINT cbOemStr;
/* length of OEM string */
/
```

The OemToAnsiBuff function translates a string from the OEM-defined character set into the Windows character set.

Parameter	Description
lpszOemStr	Points to a buffer containing one or more characters from the OEM-defined character set.
lpszWindowsStr	Points to the location where the translated string is to be copied. To translate the string in place, the lpszWindowsStr parameter can be the same as the lpszOemStr parameter.
cbOemStr	Specifies the length, in bytes, of the buffer pointed to by lpszOemStr. If cbOemStr is 0, the length is 64K.

### Returns

This function does not return a value.

### See Also

[AnsiToOem](#), [OemToAnsi](#)

## OpenFile (2.x)

```
HFILE OpenFile(lpszFileName, lpOpenBuff, fuMode)
LPCSTR lpszFileName;          /* address of filename          */
OFSTRUCT FAR* lpOpenBuff;    /* address of buffer for file information */
/
UINT fuMode;
/* action and attributes
/
```

The OpenFile function creates, opens, reopens, or deletes a file.

Parameter	Description																												
lpszFileName	Points to a null-terminated string that names the file to be opened. The string must consist of characters from the Windows character set and cannot contain wildcards.																												
lpOpenBuff	Points to the <u>OFSTRUCT</u> structure that will receive information about the file when the file is first opened. The structure can be used in subsequent calls to the OpenFile function to refer to the open file.																												
fuMode	Specifies the action to take and the attributes for the file. This parameter can be a combination of the following values:																												
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>OF_CANCEL</u></td><td>Adds a Cancel button to the <u>OF_PROMPT</u> dialog box. Pressing the Cancel button directs OpenFile to return a file-not-found error message.</td></tr><tr><td><u>OF_CREATE</u></td><td>Creates a new file. If the file already exists, it is truncated to zero length. When this flag is specified, the sharing flags are ignored. If a file must be shared it should be closed after it is created and then reopened with the appropriate sharing flags.</td></tr><tr><td><u>OF_DELETE</u></td><td>Deletes the file.</td></tr><tr><td><u>OF_EXIST</u></td><td>Opens the file, and then closes it. This value is used to test for file existence. Using this value does not change the file date.</td></tr><tr><td><u>OF_PARSE</u></td><td>Fills the <u>OFSTRUCT</u> structure but carries out no other action.</td></tr><tr><td><u>OF_PROMPT</u></td><td>Displays a dialog box if the requested file does not exist. The dialog box informs the user that Windows cannot find the file and prompts the user to insert the file in drive A.</td></tr><tr><td><u>OF_READ</u></td><td>Opens the file for reading only.</td></tr><tr><td><u>OF_READWRITE</u></td><td>Opens the file for reading and writing.</td></tr><tr><td><u>OF_REOPEN</u></td><td>Opens the file using information in the reopen buffer.</td></tr><tr><td><u>OF_SEARCH</u></td><td>Windows searches in directories even when the file name includes a full path.</td></tr><tr><td><u>OF_SHARE_COMPAT</u></td><td>Opens the file with compatibility mode, allowing any program on a given machine to open the file any number of times. OpenFile fails if the file has been opened with any of the other sharing modes.</td></tr><tr><td><u>OF_SHARE_DENY_NONE</u></td><td>Opens the file without denying other programs read or write access to the file. OpenFile fails if the file has been opened in compatibility mode by any other program.</td></tr><tr><td><u>OF_SHARE_DENY_READ</u></td><td>Opens the file and denies other programs read access to the file. OpenFile fails if the file has</td></tr></tbody></table>	Value	Meaning	<u>OF_CANCEL</u>	Adds a Cancel button to the <u>OF_PROMPT</u> dialog box. Pressing the Cancel button directs OpenFile to return a file-not-found error message.	<u>OF_CREATE</u>	Creates a new file. If the file already exists, it is truncated to zero length. When this flag is specified, the sharing flags are ignored. If a file must be shared it should be closed after it is created and then reopened with the appropriate sharing flags.	<u>OF_DELETE</u>	Deletes the file.	<u>OF_EXIST</u>	Opens the file, and then closes it. This value is used to test for file existence. Using this value does not change the file date.	<u>OF_PARSE</u>	Fills the <u>OFSTRUCT</u> structure but carries out no other action.	<u>OF_PROMPT</u>	Displays a dialog box if the requested file does not exist. The dialog box informs the user that Windows cannot find the file and prompts the user to insert the file in drive A.	<u>OF_READ</u>	Opens the file for reading only.	<u>OF_READWRITE</u>	Opens the file for reading and writing.	<u>OF_REOPEN</u>	Opens the file using information in the reopen buffer.	<u>OF_SEARCH</u>	Windows searches in directories even when the file name includes a full path.	<u>OF_SHARE_COMPAT</u>	Opens the file with compatibility mode, allowing any program on a given machine to open the file any number of times. OpenFile fails if the file has been opened with any of the other sharing modes.	<u>OF_SHARE_DENY_NONE</u>	Opens the file without denying other programs read or write access to the file. OpenFile fails if the file has been opened in compatibility mode by any other program.	<u>OF_SHARE_DENY_READ</u>	Opens the file and denies other programs read access to the file. OpenFile fails if the file has
Value	Meaning																												
<u>OF_CANCEL</u>	Adds a Cancel button to the <u>OF_PROMPT</u> dialog box. Pressing the Cancel button directs OpenFile to return a file-not-found error message.																												
<u>OF_CREATE</u>	Creates a new file. If the file already exists, it is truncated to zero length. When this flag is specified, the sharing flags are ignored. If a file must be shared it should be closed after it is created and then reopened with the appropriate sharing flags.																												
<u>OF_DELETE</u>	Deletes the file.																												
<u>OF_EXIST</u>	Opens the file, and then closes it. This value is used to test for file existence. Using this value does not change the file date.																												
<u>OF_PARSE</u>	Fills the <u>OFSTRUCT</u> structure but carries out no other action.																												
<u>OF_PROMPT</u>	Displays a dialog box if the requested file does not exist. The dialog box informs the user that Windows cannot find the file and prompts the user to insert the file in drive A.																												
<u>OF_READ</u>	Opens the file for reading only.																												
<u>OF_READWRITE</u>	Opens the file for reading and writing.																												
<u>OF_REOPEN</u>	Opens the file using information in the reopen buffer.																												
<u>OF_SEARCH</u>	Windows searches in directories even when the file name includes a full path.																												
<u>OF_SHARE_COMPAT</u>	Opens the file with compatibility mode, allowing any program on a given machine to open the file any number of times. OpenFile fails if the file has been opened with any of the other sharing modes.																												
<u>OF_SHARE_DENY_NONE</u>	Opens the file without denying other programs read or write access to the file. OpenFile fails if the file has been opened in compatibility mode by any other program.																												
<u>OF_SHARE_DENY_READ</u>	Opens the file and denies other programs read access to the file. OpenFile fails if the file has																												

OF\_SHARE\_DENY\_WRITE

been opened in compatibility mode or for read access by any other program.

Opens the file and denies other programs write access to the file. OpenFile fails if the file has been opened in compatibility or for write access by any other program.

OF\_SHARE\_EXCLUSIVE

Opens the file with exclusive mode, denying other programs both read and write access to the file. OpenFile fails if the file has been opened in any other mode for read or write access, even by the current program.

OF\_VERIFY

Compares the time and date in the OF\_STRUCT with the time and date of the specified file. The function returns HFILE\_ERROR if the dates and times do not agree.

OF\_WRITE

Opens the file for writing only.

#### Returns

The return value is an MS-DOS file handle if the function is successful. (This handle is not necessarily valid; for example, if the fuMode parameter is OF\_EXIST, the handle does not identify an open file, and if the fuMode parameter is OF\_DELETE, the handle is invalid.) The return value is HFILE\_ERROR if an error occurs.

#### Comments

If the lpszFileName parameter specifies a filename and extension only (or if the OF\_SEARCH flag is specified), the OpenFile function searches for a matching file in the following directories (in this order):

- 1 The current directory.
- 2 The Windows directory (the directory containing WIN.COM), whose path the GetWindowsDirectory function retrieves.
- 3 The Windows system directory (the directory containing such system files as GDI.EXE), whose path the GetSystemDirectory function retrieves.
- 4 The directory containing the executable file for the current task; the GetModuleFileName function obtains the path of this directory.
- 5 The directories listed in the PATH environment variable.
- 6 The list of directories mapped in a network.

To close the file after use, the application should call the \_lclose function.

#### See Also

GetSystemDirectory, GetWindowsDirectory, OFSTRUCT

OF\_CANCEL 0x0800

Adds a Cancel button to the OF\_PROMPT dialog box. Pressing the Cancel button directs OpenFile to return a file-not-found error message.

OF\_CANCEL 0x0800



OF\_CREATE 0x1000

Creates a new file. If the file already exists, it is truncated to zero length. When this flag is specified, the sharing flags are ignored. If a file must be shared it should be closed after it is created and then reopened with the appropriate sharing flags.

OF\_CREATE 0x1000

OF\_DELETE 0x0200

Deletes the file.

OF\_DELETE 0x0200

OF\_EXIST 0x4000

Opens the file, and then closes it. This value is used to test for file existence. Using this value does not change the file date.

OF\_EXIST 0x4000

OF\_PARSE 0x0100

Fills the OFSTRUCT structure but carries out no other action.

OF\_PARSE 0x0100



OF\_PROMPT 0x2000

Displays a dialog box if the requested file does not exist. The dialog box informs the user that Windows cannot find the file and prompts the user to insert the file in drive A.

OF\_PROMPT 0x2000

OF\_READ 0x0000

Opens the file for reading only.

OF\_READ 0x0000

OF\_READWRITE 0x0002

Opens the file for reading and writing.

OF\_READWRITE 0x0002

OF\_REOPEN 0x8000

Opens the file using information in the reopen buffer.

OF\_REOPEN 0x8000



OF\_SEARCH 0x0400

Windows searches in directories even when the file name includes a full path.

OF\_SEARCH 0x0400

OF\_SHARE\_COMPAT 0x0000

Opens the file with compatibility mode, allowing any program on a given machine to open the file any number of times. OpenFile fails if the file has been opened with any of the other sharing modes.

OF\_SHARE\_COMPAT 0x0000

OF\_SHARE\_DENY\_NONE 0x0040

Opens the file without denying other programs read or write access to the file. OpenFile fails if the file has been opened in compatibility mode by any other program.

OF\_SHARE\_DENY\_NONE 0x0040

OF\_SHARE\_DENY\_READ 0x0030

Opens the file and denies other programs read access to the file. OpenFile fails if the file has been opened in compatibility mode or for read access by any other program.

OF\_SHARE\_DENY\_READ 0x0030



OF\_SHARE\_DENY\_WRITE 0x0020

Opens the file and denies other programs write access to the file. OpenFile fails if the file has been opened in compatibility or for write access by any other program.

OF\_SHARE\_DENY\_WRITE 0x0020

OF\_SHARE\_EXCLUSIVE 0x0010

Opens the file with exclusive mode, denying other programs both read and write access to the file.

OpenFile fails if the file has been opened in any other mode for read or write access, even by the current program.

OF\_SHARE\_EXCLUSIVE 0x0010

OF\_VERIFY 0x0400

Compares the time and date in the OF\_STRUCT with the time and date of the specified file. The function returns HFILE\_ERROR if the dates and times do not agree.

OF\_VERIFY 0x0400

OF\_WRITE 0x0001

Opens the file for writing only.

OF\_WRITE 0x0001



## OpenSound (2.x)

int OpenSound(void)

This function is obsolete. Use the Windows multimedia audio functions instead. For information about these functions, see the [Microsoft Windows Multimedia Programmer's Reference](#).

■ **OutputDebugString (3.0)**

```
void OutputDebugString(lpszOutputString)
LPCSTR lpszOutputString;      /* address of string to display */
```

The OutputDebugString function displays the specified character string on the debugging terminal if a debugger is running.

Parameter	Description
lpszOutputString	Points to a null-terminated string to be displayed.

Returns  
This function does not return a value.

Comments  
This function preserves all registers.

Example  
The following example uses the OutputDebugString function to display information on the debugging terminal:

```
OutputDebugString( "\n\rcalling ValidateCodeSegments" );
ValidateCodeSegments ( ) ;
OutputDebugString( "\n\rdone" );
```

See Also  
DebugOutput

#### Correction

The previous description of this function indicated that it worked only with the debugging version of Windows. `OutputDebugString` works with both the debugging and retail version of Windows.

## PrestoChangoSelector (3.0)

```
UINT PrestoChangoSelector(uSourceSelector, uDestSelector)
```

```
UINT uSourceSelector;          /* selector to convert
```

```
*/
```

```
UINT uDestSelector;
```

```
* converted selector (allocated by AllocSelector)
```

```
/
```

The PrestoChangoSelector function generates a code selector that corresponds to a given data selector, or it generates a data selector that corresponds to a given code selector.

An application should not use this function unless it is absolutely necessary, because its use violates preferred Windows programming practices.

Parameter	Description
uSourceSelector	Specifies the selector to be converted.
uDestSelector	Specifies a selector previously allocated by the <u>AllocSelector</u> function. This previously allocated selector receives the converted selector.

### Returns

The return value is the copied and converted selector if the function is successful. Otherwise, it is zero.

### Comments

Windows does not track changes to the source selector. Consequently, before any memory can be moved, the application should use the converted destination selector immediately after it is returned by this function.

The PrestoChangoSelector function modifies the destination selector to have the same properties as the source selector, but with the opposite code or data attribute. This function changes only the attributes of the selector, not the value of the selector.

This function was named ChangeSelector in the Windows 3.0 documentation.

### See Also

AllocDStoCSAlias, AllocSelector

## ProfClear (3.0)

void ProfClear(void)

The ProfClear function discards all Microsoft Windows Profiler samples currently in the sampling buffer.

### Returns

This function does not return a value.

### Example

The following example uses the ProfClear function to clear the Profiler sampling buffer before changing the sampling rate:

```
ProfClear(); /* clears existing buffer */  
ProfSampleRate(5, 1); /* changes sampling rate */
```

## ProfFinish (3.0)

void ProfFinish(void)

The ProfFinish function stops Microsoft Windows Profiler sampling and flushes the output buffer to disk.

### Returns

This function does not return a value.

### Comments

If Profiler is running in 386 enhanced mode, the ProfFinish function also frees the buffer for system use.

### Example

The following example uses the ProfFinish function to stop sampling and flush the output buffer during WM\_DESTROY message processing:

```
case WM_DESTROY:  
    ProfFinish();  
    PostQuitMessage(0);  
    break;
```

## ProfFlush (3.0)

void ProfFlush(void)

The ProfFlush function flushes the Microsoft Windows Profiler sampling buffer to disk.

### Returns

This function does not return a value.

### Comments

Excessive use of the ProfFlush function can seriously impair application performance. An application should not use ProfFlush when MS-DOS may be unstable (inside an interrupt handler, for example).

### Example

The following example uses the ProfFlush function to flush the Profiler buffer before changing the buffer size:

```
ProfFlush(); /* flushes existing buffer */  
ProfSetup(1024, 0); /* uses a 1024K buffer*/
```

## ProfInsChk (3.0)

int ProfInsChk(void)

The ProfInsChk function determines whether Microsoft Windows Profiler is installed.

### Returns

The return value is 1 if Profiler is installed for a mode other than 386 enhanced mode, or it is 2 if Profiler is installed for 386 enhanced mode. Otherwise, the return value is 0, indicating that Profiler is not installed.

### Example

The following example uses the ProfInsChk function to determine whether the Profiler is installed:

```
int ick;
char szMsg[80];

if ((ick = ProfInsChk()) == 0)
    MessageBox(hwnd, "Profiler is not installed!",
    "ProfInsChk", MB_ICONSTOP);
else {
    strcpy(szMsg, "Profiler is installed");
    if (ick == 2) {
        strcat(szMsg, " in 386 enhanced mode");
        ProfSetup(128, 0); /* uses a 128K buffer */
    }
    MessageBox(hwnd, szMsg, "ProfInsChk", MB_OK);
}
```



### ProfSampRate (3.0)

```
void ProfSampRate(nRate286, nRate386)
int nRate286;      /* sample rate for non-386 enhanced mode */
int nRate386;     /* sample rate
for 386 enhanced mode
/
```

The ProfSampRate function sets the Microsoft Windows Profiler code-sampling rate.

Parameter	Description																												
nRate286	Specifies the sampling rate if the application is not running in 386 enhanced mode. The nRate286 parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Sampling rate</th></tr></thead><tbody><tr><td>1</td><td>122.070 microseconds</td></tr><tr><td>2</td><td>244.141 microseconds</td></tr><tr><td>3</td><td>488.281 microseconds</td></tr><tr><td>4</td><td>976.562 microseconds</td></tr><tr><td>5</td><td>1.953125 milliseconds</td></tr><tr><td>6</td><td>3.90625 milliseconds</td></tr><tr><td>7</td><td>7.8125 milliseconds</td></tr><tr><td>8</td><td>15.625 milliseconds</td></tr><tr><td>9</td><td>31.25 milliseconds</td></tr><tr><td>10</td><td>62.5 milliseconds</td></tr><tr><td>11</td><td>125 milliseconds</td></tr><tr><td>12</td><td>250 milliseconds</td></tr><tr><td>13</td><td>500 milliseconds</td></tr></tbody></table>	Value	Sampling rate	1	122.070 microseconds	2	244.141 microseconds	3	488.281 microseconds	4	976.562 microseconds	5	1.953125 milliseconds	6	3.90625 milliseconds	7	7.8125 milliseconds	8	15.625 milliseconds	9	31.25 milliseconds	10	62.5 milliseconds	11	125 milliseconds	12	250 milliseconds	13	500 milliseconds
Value	Sampling rate																												
1	122.070 microseconds																												
2	244.141 microseconds																												
3	488.281 microseconds																												
4	976.562 microseconds																												
5	1.953125 milliseconds																												
6	3.90625 milliseconds																												
7	7.8125 milliseconds																												
8	15.625 milliseconds																												
9	31.25 milliseconds																												
10	62.5 milliseconds																												
11	125 milliseconds																												
12	250 milliseconds																												
13	500 milliseconds																												
nRate386	Specifies the sampling rate, in milliseconds if the application is running in 386 enhanced mode. This value is in the range 1 through 1000.																												

#### Returns

This function does not return a value.

#### Comments

Only the rate parameter appropriate to the current mode is used; the other parameter is ignored.

The default rate is 2 milliseconds in 386 enhanced mode; in any other mode, the value is 5, which specifies a rate of 1.953125 milliseconds.

#### Example

The following example uses the ProfSampRate function to change the Profiler sampling rate to 1 millisecond in 386 enhanced mode:

```
ProfClear(); /* clears existing buffer */
ProfSampRate(5, 1); /* changes sampling rate */
```

## ProfSetup (3.0)

```
void ProfSetup(nBufferKB, nSamplesKB)
int nBufferKB;      /* size of output buffer */
int nSamplesKB;    /* amount of
sample data written to disk
/
```

The ProfSetup function specifies the size of the Microsoft Windows Profiler output buffer and how much sampling data Profiler is to write to the disk.

Profiler ignores the ProfSetup function when running with Windows in any mode other than 386 enhanced mode.

Parameter	Description
nBufferKB	Specifies the size, in kilobytes, of the output buffer. This value is in the range 1 through 1064. The default value is 64.
nSamplesKB	Specifies the amount, in kilobytes, of sampling data Profiler writes to the disk. A value of zero (the default value) specifies unlimited sampling data.

### Returns

This function does not return a value.

### Comments

Do not call the ProfSetup function after calling ProfStart. To resize memory after ProfStart has been called, first call the ProfStop function.

### Example

The following example uses the ProfSetup function to set the output buffer size to 128K if Profiler is installed in 386 enhanced mode:

```
int ick;
char szMsg[80];

if ((ick = ProfInsChk()) == 0)
    MessageBox(hwnd, "Profiler is not installed!",
    "ProfInsChk", MB_ICONSTOP);
else {
    strcpy(szMsg, "Profiler is installed");
    if (ick == 2) {
        strcat(szMsg, " in 386 enhanced mode");
        ProfSetup(128, 0); /* uses a 128K buffer */
    }
    MessageBox(hwnd, szMsg, "ProfInsChk", MB_OK);
}
```

### See Also

ProfStart, ProfStop

## ProfStart (3.0)

void ProfStart(void)

The ProfStart function starts Microsoft Windows Profiler sampling.

### Returns

This function does not return a value.

### Example

The following example uses the ProfStart and ProfStop functions to sample during the message-queue dispatch process:

```
/* Acquire and dispatch messages until WM_QUIT is received. */
while (GetMessage(&msg, /* message structure */
    (HWND) NULL, /* handle of window receiving message */
    0, /* lowest message to examine*/
    0)) /* highest message to examine */
{
    ProfStart();

    TranslateMessage(&msg); /* translates virtual-key codes */
    DispatchMessage(&msg); /* dispatches message to window */

    ProfStop();
}
```

See Also

ProfStop

## ProfStop (3.0)

void ProfStop(void)

The ProfStop function stops Microsoft Windows Profiler sampling.

### Returns

This function does not return a value.

### Example

The following example uses the ProfStart and ProfStop functions to sample during the message-queue dispatch process:

```
/* Acquire and dispatch messages until WM_QUIT is received. */  
  
while (GetMessage(&msg, /* message structure */  
    (HWND) NULL, /* handle of window receiving message */  
    0, /* lowest message to examine*/  
    0)) /* highest message to examine */  
    {  
        ProfStart();  
  
        TranslateMessage(&msg); /* translates virtual-key codes */  
        DispatchMessage(&msg); /* dispatches message to window */  
  
        ProfStop();  
    }
```

See Also

ProfStart

## SetErrorMode (2.x)

```
UINT SetErrorMode(fuErrorMode)
UINT fuErrorMode;    /* specifies the error-mode flag */
```

The SetErrorMode function controls whether Windows handles MS-DOS Interrupt 24h errors or allows the calling application to handle them.

Parameter	Description								
fuErrorMode	Specifies the error-mode flag. The flag can be a combination of the following values:								
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>SEM_FAILCRITICALERRORS</u></td><td>Windows does not display the critical-error-handler message box and returns the error to the calling application.</td></tr><tr><td><u>SEM_NOGPFAULTERRORBOX</u></td><td>Windows does not display the general-protection-fault message box. This flag should be set only by debugging applications that handle GP faults themselves.</td></tr><tr><td><u>SEM_NOOPENFILEERRORBOX</u></td><td>Windows does not display a message box when it fails to find a file.</td></tr></tbody></table>	Value	Meaning	<u>SEM_FAILCRITICALERRORS</u>	Windows does not display the critical-error-handler message box and returns the error to the calling application.	<u>SEM_NOGPFAULTERRORBOX</u>	Windows does not display the general-protection-fault message box. This flag should be set only by debugging applications that handle GP faults themselves.	<u>SEM_NOOPENFILEERRORBOX</u>	Windows does not display a message box when it fails to find a file.
Value	Meaning								
<u>SEM_FAILCRITICALERRORS</u>	Windows does not display the critical-error-handler message box and returns the error to the calling application.								
<u>SEM_NOGPFAULTERRORBOX</u>	Windows does not display the general-protection-fault message box. This flag should be set only by debugging applications that handle GP faults themselves.								
<u>SEM_NOOPENFILEERRORBOX</u>	Windows does not display a message box when it fails to find a file.								

### Returns

The return value is the previous state of the error-mode flag, if the function is successful.

### Example

The following example uses the SetErrorMode function to turn off the file-not-found message box (the application handles this error itself):

```
/* Turn off the "File not found" error box. */
SetErrorMode(SEM_NOOPENFILEERRORBOX);

/* Load the TOOLHELP.DLL library module. */
hinstToolHelp = LoadLibrary("TOOLHELP.DLL");

if (hinstToolHelp > HINSTANCE_ERROR) {    /* loaded successfully */
    .
    . /* Use the DLL here. */
    .
}
else {
    strcpy(szBuf, "LoadLibrary failed");
}

MessageBox(NULL, szBuf, "Library Functions", MB_ICONHAND);
```

SEM\_FAILCRITICALERRORS 0x0001

Windows does not display the critical-error-handler message box and returns the error to the calling application.

SEM\_FAILCRITICALERRORS 0x0001

SEM\_NOGPFALTERRORBOX 0x0002

Windows does not display the general-protection-fault message box. This flag should be set only by debugging applications that handle GP faults themselves.



SEM\_NOGPFAULTERRORBOX 0x0002

SEM\_NOOPENFILEERRORBOX 0x8000

Windows does not display a message box when it fails to find a file.

SEM\_NOOPENFILEERRORBOX 0x8000

## SetHandleCount (3.0)

```
UINT SetHandleCount(cHandles)
UINT cHandles;          /* number of file handles needed */
```

The SetHandleCount function changes the number of file handles available to a task.

Parameter	Description
cHandles	Specifies the number of file handles the application requires. This count cannot be greater than 255.

### Returns

The return value is the number of file handles available to the application, if the function is successful. This number may be less than the number of handles specified.

### Comments

By default, the maximum number of file handles available to a task is 20.

### Example

The following example uses the SetHandleCount function to set the number of available file handles to 30:

```
UINT cHandles;
char szBuf[80];

cHandles = SetHandleCount(30);

sprintf(szBuf, "%d handles available", cHandles);
MessageBox(hwnd, szBuf, "SetHandleCount", MB_OK);
```

■ **SetResourceHandler (2.x)**

```
RSRCHDLRPROC SetResourceHandler(hinst, lpszType, lpLoadProc)
HINSTANCE hinst;                /* handle of application instance */
LPCSTR lpszType;
/* address of resource-type identifier
/
RSRCHDLRPROC lpLoadProc;
/* callback procedure-instance address
/
```

The SetResourceHandler function installs a callback function that loads resources.

Parameter	Description
hinst	Identifies the instance of the module whose executable file contains the resource.
lpszType	Points to a null-terminated string that specifies a resource type. For predefined resource types, the high-order word should be zero and the low-order word should indicate the resource type.
lpLoadProc	Specifies the procedure-instance address of the application-supplied callback function. For more information, see the description of the <u>LoadProc</u> callback function.

**Returns**

The return value is a pointer to the previously installed resource handler, if the function is successful. If no resource handler has been explicitly installed, the return value is a pointer to the default resource handler.

**Comments**

An application may find this function useful for handling its own resource types, but the use of this function is not required.

The address passed as the lpLoadProc parameter must be created by using the MakeProcInstance function.

**See Also**

FindResource, LoadProc, LockResource, MakeProcInstance

### Correction

The second parameter points to a null-terminated string that specifies the resource type. Previous documentation stated that it pointed to a short integer.

### SetSelectorBase (3.1)

```
UINT SetSelectorBase(selector, dwBase)
UINT selector;          /* selector to modify */
DWORD dwBase;          /* new base */
```

The SetSelectorBase function sets the base address of a selector.

Parameter	Description
selector	Specifies the selector value to modify.
dwBase	Specifies the new base value. This value is the starting linear address that selector will reference.

#### Returns

The return value is the selector value, if the function is successful. If an error occurred, the return value is zero.

#### Comments

Because this function is selector-based, it will not exist in the Win32 API.

#### See Also

[GetSelectorBase](#), [GetSelectorLimit](#), [SetSelectorLimit](#)

### SetSelectorLimit (3.1)

```
UINT SetSelectorLimit(selector, dwLimit)
UINT selector;          /* selector to modify */
DWORD dwLimit;         /* new limit */
```

The SetSelectorLimit function sets the limit of a selector.

Parameter	Description
selector	Specifies the selector to modify.
dwLimit	Specifies the new limit value for selector. On a 80286, this value must be less than 0x10000.

#### Returns

The return value is always zero.

#### Comments

Because this function is selector-based, it will not exist in the Win32 API.

#### See Also

[GetSelectorBase](#), [GetSelectorLimit](#), [SetSelectorBase](#)



## SetSoundNoise (2.x)

```
int SetSoundNoise(fnSource, nDuration)
int fnSource;      /* source of noise */
int nDuration;     /* duration of noise */
```

This function is obsolete. Use the Microsoft Windows multimedia audio functions instead. For information about audio functions, see the Microsoft Windows Multimedia Programmer's Reference.

## SetSwapAreaSize (2.x)

```
LONG SetSwapAreaSize(cCodeParagraphs)
UINT cCodeParagraphs;          /* number of paragraphs for code */
```

The SetSwapAreaSize function sets the amount of memory that an application uses for its code segments.

Parameter	Description
cCodeParagraphs	Specifies the number of 16-byte paragraphs requested by the application for use as code segments. If this parameter is zero, the return value specifies the current size of the code-segment space.

### Returns

The return value is the amount of space available for the code segment, if the function is successful. The low-order word specifies the number of paragraphs obtained for use as a code-segment space (or the current size if the cCodeParagraphs parameter is zero); the high-order word specifies the maximum size available.

### Comments

If cCodeParagraphs specifies a size larger than is available, this function sets the size to the available amount. The maximum amount of memory available is one half the space remaining after Windows is loaded.

Calling this function can improve an application's performance by preventing Windows from swapping code segments to the hard disk. However, increasing the code-segment space reduces the amount of memory available for data objects and can reduce the performance of other applications.

### See Also

[GetNumTasks](#), [GlobalAlloc](#)

## SetVoiceAccent (2.x)

```
int SetVoiceAccent(nVoice, nTempo, nVolume, fnMode, nPitch)
int nVoice;          /* voice queue */
int nTempo;          /* number of quarter
notes per minute */
/
int nVolume;
* volume level
/
int fnMode;
* how notes are to be played
/
int nPitch;
* pitch
/
```

This function is obsolete. Use the Microsoft Windows multimedia audio functions instead. For information about these functions, see the Microsoft Windows Multimedia Programmer's Reference.

## SetVoiceEnvelope (2.x)

```
int SetVoiceEnvelope(nVoice, nShape, nRepeat)
int nVoice;          /* voice queue          */
int nShape;          /* index into an OEM */
wave-shape table
/
int nRepeat;
* repetition count
/
```

This function is obsolete. Use the Microsoft Windows multimedia audio functions instead. For information about these functions, see the Microsoft Windows Multimedia Programmer's Reference.

## SetVoiceNote (2.x)

```
int SetVoiceNote(voice, value, length, cdots)
int voice;      /* voice queue */
int value;      /* note */
int length;     /* length of note */
int cdots;     /
* duration of note
/
```

This function is obsolete. Use the Microsoft Windows multimedia audio functions instead. For information about these functions, see the Microsoft Windows Multimedia Programmer's Reference.

## SetVoiceQueueSize (2.x)

```
int SetVoiceQueueSize(nVoice, cbQueue)
int nVoice;          /* voice queue */
int cbQueue;         /* size of queue */
```

This function is obsolete. Use the Microsoft Windows multimedia audio functions instead. For information about these functions, see the Microsoft Windows Multimedia Programmer's Reference.

## SetVoiceSound (2.x)

```
int SetVoiceSound(nVoice, dwFrequency, nDuration)
int nVoice;          /* voice queue */
DWORD dwFrequency;  /* frequency */
int nDuration;      /* duration of sound */
/
```

This function is obsolete. Use the Microsoft Windows multimedia audio functions instead. For information about these functions, see the Microsoft Windows Multimedia Programmer's Reference.

## SetVoiceThreshold (2.x)

```
int SetVoiceThreshold(voice, cNotesThreshold)
int voice;           /* voice queue */
int cNotesThreshold;           /* threshold level */
```

This function is obsolete. Use the Microsoft Windows multimedia audio functions instead. For information about these functions, see the Microsoft Windows Multimedia Programmer's Reference.



## SetWinDebugInfo (3.1)

```
BOOL SetWinDebugInfo(lpwdi)
const WINDEBDEBUGINFO FAR* lpwdi;          /* address of WINDEBDEBUGINFO structure */
```

The SetWinDebugInfo function sets current system-debugging information for the debugging version of the Windows 3.1 operating system.

Parameter	Description
lpwdi	Points to a <u>WINDEBDEBUGINFO</u> structure that specifies the type of debugging information to be set.

### Returns

The return value is nonzero if the function is successful. It is zero if the pointer specified in the lpwdi parameter is invalid, the flags member of the WINDEBDEBUGINFO structure is invalid, or the function is not called in the debugging version of Windows 3.1.

### Comments

The flags member of the WINDEBDEBUGINFO structure specifies which debugging information should be set. Applications need initialize only those members of the WINDEBDEBUGINFO structure that correspond to the flags set in the flags member.

Changes to debugging information made by calling SetWinDebugInfo apply only until you exit the system or restart your computer.

### See Also

GetWinDebugInfo, WINDEBDEBUGINFO

■ **SizeofResource (2.x)**

```
DWORD SizeofResource(hinst, hrsrc)
HINSTANCE hinst;          /* handle of module with resource */
HRSRC hrsrc;              /* handle of
resource
/
```

The SizeofResource function returns the size, in bytes, of the given resource.

Parameter	Description
hinst	Identifies the instance of the module whose executable file contains the resource.
hrsrc	Identifies the resource. This handle must have been created by using the <u>FindResource</u> function.

**Returns**

The return value specifies the number of bytes in the resource, if the function is successful. It is zero if the resource cannot be found.

**Comments**

The value returned may be larger than the resource due to alignment. An application should not rely upon this value for the exact size of a resource.

**See Also**

AccessResource, FindResource

Windows 3.1 changes

The return value is now a DWORD instead of a WORD.

**StartSound (2.x)**

`int StartSound(void)`

This function is obsolete. Use the Microsoft Windows multimedia audio functions instead. For information about these functions, see the [Microsoft Windows Multimedia Programmer's Reference](#).

StopSound (2.x)

int StopSound(void)

This function is obsolete. Use the Microsoft Windows multimedia audio functions instead. For information about these functions, see the Microsoft Windows Multimedia Programmer's Reference.

## SwapRecording (3.0)

```
void SwapRecording(fuFlag)
UINT fuFlag;      /* whether to start or stop swap recording */
```

The SwapRecording function starts or stops recording data about memory swapping. Because this function can be used only in real mode, it cannot be used with Windows 3.1.

## SwitchStackBack (3.0)

void SwitchStackBack(void)

The SwitchStackBack function restores the stack of the current task, canceling the effect of the SwitchStackTo function.

### Returns

This function does not return a value.

### Comments

SwitchStackBack preserves the contents of the AX:DX registers when it returns.

### See Also

SwitchStackTo

### SwitchStackTo (3.0)

```
void SwitchStackTo(uStackSegment, uStackPointer, uStackTop)
UINT uStackSegment;          /* new stack data segment */
UINT uStackPointer;          /* offset of
beginning of stack
/
UINT uStackTop;
* offset of top of stack
/
```

The SwitchStackTo function changes the stack of the current task to the specified data segment.

Parameter	Description
uStackSegment	Specifies the data segment to contain the stack.
uStackPointer	Specifies the offset to the beginning of the stack in the data segment.
uStackTop	Specifies the offset to the top of the stack from the beginning of the stack.

#### Returns

This function does not return a value.

#### Comments

Dynamic-link libraries (DLLs) do not have private stacks; instead, a DLL uses the stack of the task that calls the library. As a result, a DLL function fails if it treats the contents of the data-segment (DS) and stack-segment (SS) registers as equal. A task can call SwitchStackTo before calling a function in a DLL that treats the SS and DS registers as equal. When the DLL function returns, the task must then call the SwitchStackBack function to redirect its stack to its own data segment.

A DLL can also call SwitchStackTo before calling a function that assumes SS and DS to be equal and then call SwitchStackBack before returning to the task that called the DLL function.

Calls to SwitchStackTo and SwitchStackBack cannot be nested. That is, after calling SwitchStackTo, an application must call SwitchStackBack before calling SwitchStackTo again.

#### See Also

SwitchStackBack



SyncAllVoices (2.x)

int SyncAllVoices(void)

This function is obsolete. Use the Microsoft Windows multimedia audio functions instead. For information about these functions, see the Microsoft Windows Multimedia Programmer's Reference.

## Throw (2.x)

```
void Throw(lpCatchBuf, nErrorReturn)
const int FAR* lpCatchBuf;      /* address of CATCHBUF saved by Catch */
int nErrorReturn;              /* value to return from Catch function */
```

The Throw function restores the execution environment to the values saved in the specified array. Execution then transfers to the Catch function that copied the environment to the array.

Parameter	Description
lpCatchBuf	Points to a <u>CATCHBUF</u> array that contains the execution environment. This array must have been set by a previous call to the <u>Catch</u> function.
nErrorReturn	Specifies the value to be returned to the <u>Catch</u> function. The meaning of the value is determined by the application. The value should be nonzero, so that the call to the <u>Catch</u> function can distinguish between a return from <u>Catch</u> (which returns zero) and a return from <u>Throw</u> .

### Returns

This function does not return a value.

### Comments

The Throw function is similar to the C run-time function longjmp.

The function that calls Catch must free any resources allocated between the time Catch was called and the time Throw was called.

Do not use the Throw function across messages. For example, if an application calls Catch while processing a WM\_CREATE message and then calls Throw while processing a WM\_PAINT message, the application will terminate.

### Example

The following example calls the Catch function to save the current execution environment before calling a recursive sort function. The first return from Catch is zero. If the doSort function calls the Throw function, execution will again return to the Catch function. This time, Catch returns the STACKOVERFLOW error passed by the doSort function. The doSort function is recursive--that is, it calls itself. It maintains a variable, wStackCheck, that is used to check the amount of stack space used. If more than 3K of the stack has been used, doSort calls Throw to drop out of all the nested function calls back into the function that called Catch.

```
#define STACKOVERFLOW 1

UINT uStackCheck;
CATCHBUF catchbuf;

{
    int iReturn;
    char szBuf[80];

    if ((iReturn = Catch((int FAR*) catchbuf)) != 0) {
        . /* Error processing goes here. */
        .
        }
    else {
        uStackCheck = 0; /* initializes stack-usage count */
        doSort(1, 100); /* calls sorting function */
    }
    break;
}

void doSort(int sLeft, int sRight)
{
    int sLast;
```

```

/*
 * Determine whether more than 3K of the stack has been
 * used, and if so, call Throw to drop back into the
 * original calling application.
 *
 * The stack is incremented by the size of the two parameters,
 * the two local variables, and the return value (2 for a near
 * function call).
 */

    uStackCheck += (sizeof(int) * 4) + 2;

    if (uStackCheck > (3 * 1024))
        Throw((int FAR*) catchbuf, STACKOVERFLOW);
    .
    . /* A sorting algorithm goes here. */
    .

    doSort(sLeft, sLast - 1); /* note recursive call*/
    uStackCheck -= 10; /* updates stack-check variable */
}

```

See Also

Catch

## ToAscii (3.0)

```
int ToAscii(uVirtKey, uScanCode, lpbKeyState, lpdwTransKey, fuState)
UINT uVirtKey; /* virtual-key code */
UINT uScanCode; /* scan code */
BYTE FAR* lpbKeyState;
/* address of key-state array */
DWORD FAR* lpdwTransKey;
/* 32-bit buffer for translated key */
UINT fuState;
/* active-menu flag */
```

The ToAscii function translates the specified virtual-key code and keyboard state to the corresponding Windows character or characters.

Parameter	Description
uVirtKey	Specifies the virtual-key code to be translated.
uScanCode	Specifies the hardware scan code of the key to be translated. The high-order bit of this value is set if the key is not pressed (is up).
lpbKeyState	Points to a 256-byte array that contains the current keyboard state. Each element (byte) in the array contains the state of one key. If the high-order bit of a byte is set, the key is pressed (is down).
lpdwTransKey	Points to a doubleword buffer to receive the translated Windows character or characters.
fuState	Specifies whether a menu is active. This parameter must be 1 if a menu is active, or zero otherwise.

### Returns

The return value is a negative value if the specified key is a dead key. Otherwise, it is one of the following values:

Value	Meaning
2	Two characters were copied to the buffer. This is usually an accent and a dead-key character, when the dead key cannot be translated otherwise.
1	One Windows character was copied to the buffer.
0	The specified virtual key has no translation for the current state of the keyboard.

### Comments

If a previous dead key is stored in the keyboard driver, the parameters supplied to the ToAscii function might not be sufficient to translate the virtual-key code.

Typically, ToAscii performs the translation based on the virtual-key code. In some cases, however, the uScanCode parameter may be used to distinguish between a key press and a key release. The scan code is used for translating ALT+number key combinations.

### See Also

[OemKeyScan](#), [VkKeyScan](#)

## UnlockSegment (2.x)

```
void UnlockSegment(uSegment)
UINT uSegment;      /* specifies segment to unlock */
```

The `UnlockSegment` function unlocks the specified discardable memory segment. The function decrements (decreases by one) the segment's lock count. The segment is completely unlocked and subject to discarding when the lock count reaches zero.

Parameter	Description
<code>uSegment</code>	Specifies the segment address of the segment to be unlocked. If this parameter is -1, the <code>UnlockSegment</code> function unlocks the current data segment.

### Returns

The return value is the lock count for the segment, if the function is successful. This function returns its result in the CX register. When the CX register contains zero, the segment is completely unlocked.

The value returned when the function is called in C should be ignored, because the return value can be checked only in assembly language.

### Comments

An application should not rely on the return value to determine the number of times it must subsequently call `UnlockSegment` for the segment.

Other functions also can affect the lock count of a memory object. For a list of these functions, see the description of the [GlobalFlags](#) function.

Each time an application calls [LockSegment](#) for a segment, it must eventually call `UnlockSegment` for the segment.

### See Also

[GlobalFlags](#), [LockSegment](#), [UnlockData](#)

## ValidateCodeSegments (3.0)

void ValidateCodeSegments(void)

The ValidateCodeSegments function tests all code segments for random memory overwrites. The function works only in real mode (for Windows versions earlier than 3.1) and only with the debugging version of Windows.

### Returns

This function does not return a value.

### Comments

Because code segments are not writeable in protected mode (standard or enhanced), this function does nothing in Windows 3.1.

### See Also

[ValidateFreeSpaces](#)

## ValidateFreeSpaces (2.x)

void ValidateFreeSpaces(void)

The `ValidateFreeSpaces` function checks free segments in memory for valid contents. This function is available only in the debugging version of Windows.

### Returns

This function does not return a value.

### Comments

In the debugging version of Windows, the kernel fills all the bytes in free segments with the hexadecimal value 0x0CC. This function begins checking for valid contents in the free segment with the lowest address; it continues checking until it finds an invalid byte or until it has determined that all free space contains valid contents. Before calling this function, put the following lines in the WIN.INI file:

```
[KERNEL]
EnableFreeChecking=1
EnableHeapChecking=1
```

Windows sends debugging information to the debugging terminal if an invalid byte is encountered, and then it performs a fatal exit.

The `[KERNEL]` entries in WIN.INI cause automatic checking of free memory. Before returning a memory object to the application in response to a call to the `GlobalAlloc` function, Windows checks that memory to make sure it is filled with 0x0CC. Before a call to the `GlobalCompact` function, all free memory is checked. Note that using this function slows Windows systemwide by about twenty percent.

### See Also

[GlobalAlloc](#), [GlobalCompact](#), [ValidateCodeSegments](#)

## VkKeyScan (2.x)

```
UINT VkKeyScan(uChar)
UINT uChar;      /* character to translate */
```

The `VkKeyScan` function translates a Windows character to the corresponding virtual-key code and shift state for the current keyboard.

Parameter	Description
<code>uChar</code>	Specifies the character to be translated to a virtual-key code.

### Returns

The return value is the virtual-key code and shift state, if the function is successful. The low-order byte contains the virtual-key code; the high-order byte contains the shift state, which can be one of the following:

Value	Meaning
1	Character is shifted.
2	Character is a control character.
3-5	Shift-key combination that is not used for characters.
6	Character is generated by the CTRL+ALT key combination.
7	Character is generated by the SHIFT+CTRL+ALT key combination.

If no key is found that translates to the passed Windows code, the return value is -1.

### Comments

Translations for the numeric keypad (`VK_NUMPAD0` through `VK_DIVIDE`) are ignored. This function is intended to force a translation for the main keyboard only.

Applications that send characters by using the `WM_KEYUP` and `WM_KEYDOWN` messages use this function.

### See Also

`OemKeyScan`



## WaitSoundState (2.x)

```
int WaitSoundState(fnState)
int fnState;    /* state to wait for */
```

This function is obsolete. Use the Microsoft Windows multimedia audio functions instead. For information about these functions, see the Microsoft Windows Multimedia Programmer's Reference.

## WinExec (3.0)

```
UINT WinExec(lpszCmdLine, fuCmdShow)
LPCSTR lpszCmdLine; /* address of command line */
UINT fuCmdShow; /* window state of new app. */
/
```

The WinExec function runs the specified application.

Parameter	Description
lpszCmdLine	Points to a null-terminated Windows character string that contains the command line (filename plus optional parameters) for the application to be run. If the string does not contain a path, Windows searches the directories in this order: <ol style="list-style-type: none"><li>1 The current directory.</li><li>2 The Windows directory (the directory containing WIN.COM); the <u>GetWindowsDirectory</u> function retrieves the path of this directory.</li><li>3 The Windows system directory (the directory containing such system files as GDI.EXE); the <u>GetSystemDirectory</u> function retrieves the path of this directory.</li><li>4 The directory containing the executable file for the current task; the <u>GetModuleFileName</u> function retrieves the path of this directory.</li><li>5 The directories listed in the PATH environment variable.</li><li>6 The directories mapped in a network.</li></ol>
fuCmdShow	Specifies how a Windows application window is to be shown. See the description of the <u>ShowWindow</u> function for a list of the acceptable values for the fuCmdShow parameter. For a non-Windows application, the program-information file (PIF), if any, for the application determines the window state.

### Returns

The return value identifies the instance of the loaded module, if the function is successful. Otherwise, the return value is an error value less than 32.

### Errors

The error value may be one of the following:

Value	Meaning
0	System was out of memory, executable file was corrupt, or relocations were invalid.
2	File was not found.
3	Path was not found.
5	Attempt was made to dynamically link to a task, or there was a sharing or network-protection error.
6	Library required separate data segments for each task.
8	There was insufficient memory to start the application.
10	Windows version was incorrect.
11	Executable file was invalid. Either it was not a Windows application or there was an error in the .EXE image.
12	Application was designed for a different operating system.
13	Application was designed for MS-DOS 4.0.
14	Type of executable file was unknown.
15	Attempt was made to load a real-mode application (developed for an earlier version of Windows).
16	Attempt was made to load a second instance of an executable file containing multiple data segments that were not marked read-only.
19	Attempt was made to load a compressed executable file. The file must be decompressed before it can be loaded.
20	Dynamic-link library (DLL) file was invalid. One of the DLLs required to run this application was corrupt.
21	Application requires Microsoft Windows 32-bit extensions.

## Comments

The [LoadModule](#) function provides an alternative method for running an application.

## Example

The following example uses the WinExec function to run DRAW.EXE:

```
WORD wReturn;
char szMsg[80];

wReturn = WinExec("draw", SW_SHOW);

if (wReturn < 32) {
    sprintf(szMsg, "WinExec failed; error code = %d", wReturn);
    MessageBox(hwnd, szMsg, "Error", MB_ICONSTOP);
}
else {
    sprintf(szMsg, "WinExec returned %d", wReturn);
    MessageBox(hwnd, szMsg, "", MB_OK);
}
```

## See Also

[GetModuleFileName](#), [GetSystemDirectory](#), [GetWindowsDirectory](#), [LoadModule](#), [ShowWindow](#)

### WritePrivateProfileString (3.0)

```
BOOL WritePrivateProfileString(lpszSection, lpszEntry, lpszString, lpszFilename)
LPCSTR lpszSection;          /* address of section          */
LPCSTR lpszEntry;            /* address of entry           */
/*
/
LPCSTR lpszString;          /* address of string to add   */
/
LPCSTR lpszFilename;        /* address of initialization filename
/
```

The WritePrivateProfileString function copies a character string into the specified section of the specified initialization file.

Parameter	Description
lpszSection	Points to a null-terminated string that specifies the section to which the string will be copied. If the section does not exist, it is created. The name of the section is case-independent; the string may be any combination of uppercase and lowercase letters.
lpszEntry	Points to the null-terminated string containing the entry to be associated with the string. If the entry does not exist in the specified section, it is created. If this parameter is NULL, the entire section, including all entries within the section, is deleted.
lpszString	Points to the null-terminated string to be written to the file. If this parameter is NULL, the entry specified by the lpszEntry parameter is deleted.
lpszFilename	Points to a null-terminated string that names the initialization file.

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

To improve performance, Windows keeps a cached version of the most-recently accessed initialization file. If that filename is specified and the other three parameters are NULL, Windows flushes the cache.

Sections in the initialization file have the following form:

```
[section]
entry=string
.
.
.
```

If lpszFilename does not contain a fully qualified path and filename for the file, WritePrivateProfileString searches the Windows directory for the file. If the file does not exist, this function creates the file in the Windows directory.

If lpszFilename contains a fully qualified path and filename and the file does not exist, this function creates the file. The specified directory must already exist.

An application should use a private (application-specific) initialization file to record information that affects only that application. This improves the performance of both the application and Windows itself by reducing the amount of information that Windows must read when it accesses the initialization file. The exception to this is that device drivers should use the SYSTEM.INI file, to reduce the number of initialization files Windows must open and read during the startup process.

An application can use the WriteProfileString function to add a string to the WIN.INI file.

#### Example

The following example uses the WritePrivateProfileString function to add the string "testcode.c" to the LastFile entry in the [MyApp] section of the TESTCODE.INI initialization file:

```
BOOL fSuccess;
DebugBreak( );
```

```
fSuccess = WritePrivateProfileString("MyApp",
    "LastFile", "testcode.c", "testcode.ini");

if (fSuccess)
    MessageBox(hwnd, "String added successfully",
    "WritePrivateProfileString", MB_OK);
else
    MessageBox(hwnd, "String could not be added",
    "WritePrivateProfileString", MB_ICONSTOP);
```

See Also  
WriteProfileString

## Correction

The Windows 3.0 documentation stated that this function would fail if the `lpzFilename` parameter specified a full path and the file did not exist. In fact, Windows will create the specified file. The directory where the file is to be created must exist, however; Windows will not create the directory.

## WriteProfileString (2.x)

```
BOOL WriteProfileString(lpszSection, lpszEntry, lpszString)
LPCSTR lpszSection; /* address of section */
LPCSTR lpszEntry; /* address of entry */
/* address of string to write */
```

The WriteProfileString function copies a string into the specified section of the Windows initialization file, WIN.INI.

Parameter	Description
lpszSection	Points to a null-terminated string that specifies the section to which the string is to be copied. If the section does not exist, it is created. The name of the section is case-independent; the string may be any combination of uppercase and lowercase letters.
lpszEntry	Points to the null-terminated string containing the entry to be associated with the string. If the entry does not exist in the specified section, it is created. If this parameter is NULL, the entire section, including all entries within the section, is deleted.
lpszString	Points to the null-terminated string to be written to the file. If this parameter is NULL, the entry specified by the lpszEntry parameter is deleted.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

Windows keeps a cached version of WIN.INI to improve performance. If all three parameters are NULL, Windows flushes the cache.

Sections in the WIN.INI initialization file have the following form:

```
[section]
entry=string
.
.
.
```

### Example

The following example calls the GetWindowRect function to retrieve the dimensions of the current window, converts the dimensions of a string, and writes the string to WIN.INI by using the WriteProfileString function. The next time the application is run, it could call the GetProfileString function to read the string, convert it to numbers, and pass the numbers as parameters to the CreateWindow function, thereby creating the window again with the same dimensions it had when the application terminated.

```
RECT rect;
BOOL fSuccess;
char szBuf[20];

GetWindowRect(hwnd, &rect);

sprintf(szBuf, "%u %u %u %u",
        rect.left, rect.right - rect.left,
        rect.top, rect.bottom - rect.top);

fSuccess = WriteProfileString("MySection",
                             "Window dimensions", szBuf);

if (fSuccess)
    MessageBox(hwnd, "String added successfully",
               "WriteProfileString", MB_OK);
else
    MessageBox(hwnd, "String could not be added",
```

```
"WriteProfileString", MB_ICONSTOP);
```

See Also

[GetProfileString](#), [WritePrivateProfileString](#)



## wsprintf (3.0)

```
int _cdecl wsprintf(lpszOutput, lpszFormat, ...)
LPSTR lpszOutput;          /* address of string for output */
LPSTR lpszFormat;         /* address of
format-control string */
/
```

The `wsprintf` function formats and stores a series of characters and values in a buffer. Each argument (if any) is converted according to the corresponding format specified in the format string.

Parameter	Description
<code>lpszOutput</code>	Points to a null-terminated string to receive the string formatted as specified in the <code>lpszFormat</code> parameter.
<code>lpszFormat</code>	Points to a null-terminated string that contains the format-control string. In addition to the standard ASCII characters, a format specification for each argument appears in this string. For more information about the format specification, see the following Comments section.
...	Specifies zero or more optional arguments. The number and type of the optional arguments depend on the corresponding format-control character sequences specified in the <code>lpszFormat</code> parameter.

### Returns

The return value is the number of bytes stored in the `lpszOutput` string, not counting the terminating null character, if the function is successful.

### Comments

The largest buffer that `wsprintf` can create is 1K.

Unlike most Windows functions, `wsprintf` uses the C calling convention (`_cdecl`) rather than the Pascal calling convention. As a result, the calling function must pop arguments off the stack. Also, arguments must be pushed on the stack from right to left. In C-language modules, the C compiler performs this task. (The `wvsprintf` function uses the Pascal calling convention.)

The format-control string contains format specifications that determine the output format for the arguments that follow the `lpszFormat` parameter. Format specifications always begin with a percent sign (%). If a percent sign is followed by a character that has no meaning as a format field, the character is not formatted. For example, %% produces a single percent-sign character.

The format-control string is read from left to right. When the first format specification is encountered, it causes the value of the first argument after the format-control string to be converted according to the format specification. The second format specification causes the second argument to be converted, and so on. If there are more arguments than there are format specifications, the extra arguments are ignored. The results are undefined if there are not enough arguments for all of the format specifications.

A format specification has the following form:

`%[-][#][0][width][.precision]type`

Each field of the format specification is a single character or number signifying a particular format option. The type characters, for example, determine whether the associated argument is interpreted as a character, a string, or a number. The simplest format specification contains only the percent sign and a type character (for example, %s). The optional fields (in brackets) control other aspects of the formatting. Following are the optional and required fields and their meanings:

Field	Meaning
-	Pad the output value with blanks or zeros to the right to fill the field width, aligning the output value to the left. If this field is omitted, the output value is padded to the left, aligning it to the right.
#	Prefix hexadecimal values with 0x (lowercase) or 0X (uppercase).
0	Pad the output value with zeros to fill the field width. If this field is omitted, the output value is padded with blank spaces.
width	Convert the specified minimum number of characters. The width field is a nonnegative integer. The width specification never causes a value to be truncated; if the number of characters in the output value is greater than the specified width, or if the width field is not present, all characters of the value are printed, subject to the value of the precision field.

precision Convert the specified minimum number of digits. If there are fewer digits in the argument than the specified value, the output value is padded on the left with zeros. The value is not truncated when the number of digits exceeds the specified precision. If the specified precision is zero or omitted entirely, or if the period (.) appears without a number following it, the precision is set to 1.

For strings, convert the specified maximum number of characters.

type Format the corresponding argument as a character, a string, or a number. This field may be any of the following character sequences:

Sequence	Meaning
c	Insert a single character argument. The <code>wsprintf</code> function ignores character arguments with a numeric value of zero.
d, i	Insert a signed decimal integer argument.
ld, li	Insert a long signed decimal integer argument.
u	Insert an unsigned integer argument.
lu	Insert a long unsigned integer argument.
lx, lX	Insert a long unsigned hexadecimal integer argument in lowercase or uppercase.
s	Insert a string.

See Also  
`wvsprintf`

## Yield (2.x)

void Yield(void)

The Yield function stops the current task and starts any waiting task.

### Returns

This function does not return a value.

### Comments

The Yield function should be used only when the application is guaranteed not to receive any messages.

Applications that contain windows should use a DispatchMessage, PeekMessage, or TranslateMessage loop rather than call the Yield function directly. The message-loop functions handle message synchronization properly and yield at the appropriate times.

### See Also

DirectedYield, DispatchMessage, PeekMessage, TranslateMessage

## Kernel functions (3.1)

<u>hread</u>	Reads from a file
<u>hwrite</u>	Writes to a file
<u>lclose</u>	Closes an open file
<u>lcreat</u>	Creates or opens a file
<u>llseek</u>	Repositions the file pointer
<u>lopen</u>	Opens a file
<u>lread</u>	Reads from a file
<u>lwrite</u>	Writes to a file
<u>AccessResource</u>	Opens a resource file and locates a resource
<u>AddAtom</u>	Adds a string to the local atom table
<u>AllocDStoCSAlias</u>	Translates a data segment to a code segment
<u>AllocResource</u>	Allocates memory for a resource
<u>AllocSelector</u>	Allocates a new selector
<u>AnsiToOem</u>	Translates a Windows string to an OEM string
<u>AnsiToOemBuff</u>	Translates a Windows string to an OEM string
<u>Catch</u>	Captures the current execution environment
<u>CloseSound</u>	Not used in Windows 3.1
<u>CountVoiceNotes</u>	Not used in Windows 3.1
<u>DebugBreak</u>	Causes a breakpoint exception
<u>DebugOutput</u>	Sends messages to the debugging terminal
<u>DeleteAtom</u>	Decrements the reference count of a local atom
<u>DirectedYield</u>	Forces execution of a specified task to continue
<u>DOS3Call</u>	Issues a DOS Int 21h function request
<u>FatalAppExit</u>	Terminates an application
<u>FatalExit</u>	Displays debug info after breakpoint exception
<u>FindAtom</u>	Retrieves string atom from local atom table
<u>FindResource</u>	Locates a resource in a resource file
<u>FreeLibrary</u>	Unloads a library module instance
<u>FreeModule</u>	Unloads a module instance
<u>FreeProcInstance</u>	Frees a function instance
<u>FreeResource</u>	Unloads a resource instance
<u>FreeSelector</u>	Frees an allocated selector
<u>GetAtomHandle</u>	Retrieves an atom handle
<u>GetAtomName</u>	Retrieves a local atom string
<u>GetCodeHandle</u>	Determines the location of a function
<u>GetCodeInfo</u>	Retrieves code-segment information
<u>GetCurrentPDB</u>	Returns the selector address of the current PDB
<u>GetCurrentTask</u>	Returns the current task handle
<u>GetDOSEnvironment</u>	Returns a far pointer to the current environment
<u>GetDriveType</u>	Determines the drive type
<u>GetFreeSpace</u>	Returns number of free bytes in global heap
<u>GetInstanceData</u>	Copies data from previous instance to current one
<u>GetKBCodePage</u>	Returns the current code page
<u>GetKeyboardType</u>	Retrieves keyboard information
<u>GetKeyNameText</u>	Retrieves a string representing the key name
<u>GetModuleFileName</u>	Returns the filename for a module handle
<u>GetModuleHandle</u>	Returns a module handle for a named module
<u>GetModuleUsage</u>	Returns the reference count for a module
<u>GetNumTasks</u>	Returns the current number of tasks
<u>GetPrivateProfileInt</u>	Retrieves integer value from initialization file
<u>GetPrivateProfileString</u>	Retrieves a string from an initialization file
<u>GetProcAddress</u>	Returns the address of an exported DLL function
<u>GetProfileInt</u>	Retrieves an integer value from WIN.INI
<u>GetProfileString</u>	Retrieves a string from WIN.INI
<u>GetSelectorBase</u>	Retrieves the base address of a selector
<u>GetSelectorLimit</u>	Retrieves the limit of a selector
<u>GetSystemDirectory</u>	Returns the Windows system directory
<u>GetTempDrive</u>	Returns a disk drive letter for temporary files
<u>GetTempFileName</u>	Creates a temporary filename
<u>GetThresholdEvent</u>	Not used in Windows 3.1
<u>GetThresholdStatus</u>	Not used in Windows 3.1
<u>GetVersion</u>	Returns the current DOS and Windows versions
<u>GetWinDebugInfo</u>	Retrieves current system-debugging information

<u>GetWindowsDirectory</u>	Returns the Windows directory
<u>GetWinFlags</u>	Returns the current system configuration flags
<u>GlobalAlloc</u>	Allocates memory from the global heap
<u>GlobalCompact</u>	Generates free global memory by compacting
<u>GlobalDosAlloc</u>	Allocates memory available to DOS in real mode
<u>GlobalDosFree</u>	Frees global memory allocated by GlobalDosAlloc
<u>GlobalFix</u>	Locks a global memory object in linear memory
<u>GlobalFlags</u>	Returns information about a global memory object
<u>GlobalFree</u>	Frees a global memory object
<u>GlobalHandle</u>	Retrieves a handle for a specified selector
<u>GlobalLock</u>	Locks global memory object and returns pointer
<u>GlobalLRUNewest</u>	Moves global memory object to newest LRU position
<u>GlobalLRUOldest</u>	Moves global memory object to oldest LRU position
<u>GlobalNotify</u>	Installs a notification procedure
<u>GlobalPageLock</u>	Increments global memory page-lock count
<u>GlobalPageUnlock</u>	Decrements global memory page-lock count
<u>GlobalReAlloc</u>	Changes size/attributes of global memory object
<u>GlobalSize</u>	Returns the size of a global memory object
<u>GlobalUnfix</u>	Unlocks a global memory object in linear memory
<u>GlobalUnlock</u>	Unlocks a global memory object
<u>GlobalUnWire</u>	Not used in Windows 3.1
<u>GlobalWire</u>	Not used in Windows 3.1
<u>hmemcpy</u>	Copies bytes from source to destination buffer
<u>InitAtomTable</u>	Sets the size of the local atom table
<u>IsBadCodePtr</u>	Determines whether a code pointer is valid
<u>IsBadHugeReadPtr</u>	Determines whether a huge read pointer is valid
<u>IsBadHugeWritePtr</u>	Determines whether a huge write pointer is valid
<u>IsBadReadPtr</u>	Determines whether a read pointer is valid
<u>IsBadStringPtr</u>	Determines whether a string pointer is valid
<u>IsBadWritePtr</u>	Determines whether a write pointer is valid
<u>IsDBCSLeadByte</u>	Determines whether character is DBCS lead byte
<u>IsTask</u>	Determines whether a task handle is valid
<u>LimitEmsPages</u>	Not used in Windows 3.1
<u>LoadLibrary</u>	Loads the specified library module
<u>LoadModule</u>	Loads and executes a program
<u>LoadResource</u>	Loads the specified resource in global memory
<u>LocalAlloc</u>	Allocates memory from the local heap
<u>LocalCompact</u>	Generates free local memory by compacting
<u>LocalFlags</u>	Returns local memory object information
<u>LocalFree</u>	Frees a local memory object
<u>LocalHandle</u>	Returns the handle of a local memory object
<u>LocalInit</u>	Initializes a local heap
<u>LocalLock</u>	Locks local memory object and returns pointer
<u>LocalReAlloc</u>	Changes size or attributes of local memory object
<u>LocalShrink</u>	Shrinks the specified local heap
<u>LocalSize</u>	Returns the size of a local memory object
<u>LocalUnlock</u>	Unlocks a local memory object
<u>LockResource</u>	Returns the address of a resource
<u>LockSegment</u>	Locks a discardable memory segment
<u>LogError</u>	Identifies an error message
<u>LogParamError</u>	Identifies a parameter validation error
<u>lstrcat</u>	Appends one string to another
<u>lstrcpy</u>	Copies a string to a buffer
<u>lstrcpyn</u>	Copies characters in a string to a buffer
<u>lstrlen</u>	Returns the number of characters in a string
<u>MakeProcInstance</u>	Returns address of prolog code for function
<u>MapVirtualKey</u>	Translates a virtual-key code or scan code
<u>MulDiv</u>	Multiplies two values and divides the result
<u>NetBIOSCall</u>	Issues a NETBIOS Interrupt 5Ch call
<u>OemKeyScan</u>	Maps OEM ASCII to scan codes
<u>OemToAnsi</u>	Translates an OEM string to a Windows string
<u>OemToAnsiBuff</u>	Translates an OEM string to a Windows string
<u>OpenFile</u>	Creates, opens, reopens, or deletes a file
<u>OpenSound</u>	Not used in Windows 3.1

<u>OutputDebugString</u>	Sends a character string to the debugger
<u>PresoChangeSelector</u>	Converts code or data selector
<u>ProfClear</u>	Discards all buffered Profiler samples
<u>ProfFinish</u>	Stops Profiler sampling and flushes buffer
<u>ProfFlush</u>	Flushes the Profiler sampling buffer to a disk
<u>ProfInsChk</u>	Determines whether Profiler is installed
<u>ProfSampRate</u>	Sets the Profiler sampling rate
<u>ProfSetup</u>	Sets the Profiler buffer size and sample quantity
<u>ProfStart</u>	Starts Profiler sampling
<u>ProfStop</u>	Stops Profiler sampling
<u>SetErrorMode</u>	Controls Interrupt 24h error handling
<u>SetHandleCount</u>	Changes the number of available file handles
<u>SetResourceHandler</u>	Installs a load-resource callback function
<u>SetSelectorBase</u>	Sets the base of an existing selector
<u>SetSelectorLimit</u>	Sets the limit of a selector
<u>SetSoundNoise</u>	Not used in Windows 3.1
<u>SetSwapAreaSize</u>	Sets the amount of memory used for code segments
<u>SetVoiceAccent</u>	Not used in Windows 3.1
<u>SetVoiceEnvelope</u>	Not used in Windows 3.1
<u>SetVoiceNote</u>	Not used in Windows 3.1
<u>SetVoiceQueueSize</u>	Not used in Windows 3.1
<u>SetVoiceSound</u>	Not used in Windows 3.1
<u>SetVoiceThreshold</u>	Not used in Windows 3.1
<u>SetWinDebugInfo</u>	Sets the current system-debugging information
<u>SizeofResource</u>	Returns the size of a resource
<u>StartSound</u>	Not used in Windows 3.1
<u>StopSound</u>	Not used in Windows 3.1
<u>SwapRecording</u>	Starts or stops recording of memory swapping
<u>SwitchStackBack</u>	Restores the current task stack
<u>SwitchStackTo</u>	Changes the location of the stack
<u>SyncAllVoices</u>	Not used in Windows 3.1
<u>Throw</u>	Restores the execution environment
<u>ToAscii</u>	Translates virtual-key code to Windows character
<u>UnlockSegment</u>	Unlocks a discardable memory segment
<u>ValidateCodeSegments</u>	Tests for memory overwrites
<u>ValidateFreeSpaces</u>	Checks free memory for valid contents
<u>VkKeyScan</u>	Translates Windows character to virtual-key code
<u>WaitSoundState</u>	Not used in Windows 3.1
<u>WinExec</u>	Runs a program
<u>WritePrivateProfileString</u>	Writes a string to an initialization file
<u>WriteProfileString</u>	Writes a string to WIN.INI
<u>wsprintf</u>	Formats a string
<u>Yield</u>	Stops the current task

## CopyLZFile (3.1)

```
#include lzexpand.h

LONG CopyLZFile(hfSource, hfDest)
HFILE hfSource;      /* handle of source file      */
HFILE hfDest;        /* handle of destination file */
/
```

The CopyLZFile function copies a source file to a destination file. If the source file is compressed, this function creates a decompressed destination file. If the source file is not compressed, this function duplicates the original file.

Parameter	Description
hfSource	Identifies the source file.
hfDest	Identifies the destination file.

### Returns

The return value specifies the size, in bytes, of the destination file if the function is successful. Otherwise, it is an error value less than zero; it may be one of the following:

Value	Meaning
LZERROR_BADINHANDLE	The handle identifying the source file was not valid.
LZERROR_BADOUTHANDLE	The handle identifying the destination file was not valid.
LZERROR_BADVALUE	The input parameter is out of the allowable range.
LZERROR_GLOBALLOC	There is insufficient memory for the required buffers.
LZERROR_GLOBLOCK	The handle identifying the internal data structures is invalid.
LZERROR_READ	The source file format was not valid.
LZERROR_UNKNOWNALG	The source file was compressed with an unrecognized compression algorithm.
LZERROR_WRITE	There is insufficient space for the output file.

### Comments

This function is identical to the LZCopy function.

The CopyLZFile function is designed for copying or decompressing multiple files, or both. To allocate required buffers, an application should call the LZStart function prior to calling CopyLZFile. To free these buffers, an application should call the LZDone function after copying the files.

If the function is successful, the file identified by hfDest is decompressed.

If the source or destination file is opened by using a C run-time function (rather than by using the \_lopen or OpenFile function), it must be opened in binary mode.

### Example

The following example uses the CopyLZFile function to create copies of four text files:

```
#define STRICT

#include <windows.h>
#include <lzexpand.h>

#define NUM_FILES 4

char *szSrc[NUM_FILES] =
    {"readme.txt", "data.txt", "update.txt", "list.txt"};
char *szDest[NUM_FILES] =
    {"readme.bak", "data.bak", "update.bak", "list.bak"};
OFSTRUCT ofStrSrc;
OFSTRUCT ofStrDest;
HFILE hfSrcFile, hfDstFile;
int i;

/* Allocate internal buffers for the CopyLZFile function. */
```

LZStart();

/\* Open, copy, and then close the files. \*/

```
for (i = 0; i < NUM_FILES; i++) {  
    hfSrcFile = LZOpenFile(szSrc[i], &ofStrSrc, OF_READ);  
    hfDstFile = LZOpenFile(szDest[i], &ofStrDest, OF_CREATE);  
    CopyLZFile(hfSrcFile, hfDstFile);  
    LZClose(hfSrcFile);  
    LZClose(hfDstFile);  
}
```

LZDone(); /\* free the internal buffers \*/

See Also

\_lopen, LZCopy, LZDone, LZStart, OpenFile



### GetExpandedName (3.1)

```
#include lzexpand.h
```

```
int GetExpandedName(lpszSource, lpszBuffer)
LPCSTR lpszSource;          /* specifies name of compressed file      */
LPSTR lpszBuffer;          /* points to buffer receiving original filename */
/
```

The GetExpandedName function retrieves the original name of a compressed file if the file was compressed with the COMPRESS.EXE utility and the /r option was specified.

Parameter	Description
lpszSource	Points to a string that specifies the name of a compressed file.
lpszBuffer	Points to a buffer that receives the name of the compressed file.

#### Returns

The return value is TRUE if the function is successful. Otherwise, it is an error value that is less than zero, and it may be LZERROR\_BADINHANDLE, which means that the handle identifying the source file was not valid.

#### Example

The following example uses the GetExpandedName function to retrieve the original filename of a compressed file:

```
char szSrc[] = {"readme.cmp"};
char szFileName[128];
OFSTRUCT ofStrSrc;
OFSTRUCT ofStrDest;
HFILE hfSrcFile, hfDstFile, hfCompFile;
int cbRead;
BYTE abBuf[512];

/* Open the compressed source file. */

hfSrcFile = OpenFile(szSrc, &ofStrSrc, OF_READ);

/*
 * Initialize internal data structures for the decompression
 * operation.
 */

hfCompFile = LZInit(hfSrcFile);

/* Retrieve the original name for the compressed file. */
GetExpandedName(szSrc, szFileName);

/* Create the destination file using the original name. */
hfDstFile = LZOpenFile(szFileName, &ofStrDest, OF_CREATE);

/* Copy the compressed source file to the destination file. */
do {
    if ((cbRead = LZRead(hfCompFile, abBuf, sizeof(abBuf))) > 0)
        write(hfDstFile, abBuf, cbRead);
    else {
        . /* handle error condition */
        .
    }
} while (cbRead == sizeof(abBuf));
```

```
/* Close the files. */
```

```
LZClose(hfSrcFile);
```

```
LZClose(hfDstFile);
```

#### Comments

This function retrieves the original filename from the header of the compressed file. If the source file is not compressed, the filename to which lpszSource points is copied to the buffer to which lpszBuffer points.

If the /r option was not set when the file was compressed, the string in the buffer to which lpszBuffer points is invalid.

## LZClose (3.1)

#include lzexpand.h

```
void LZClose(hf)
HFILE hf;      /* handle of file to be closed */
```

The LZClose function closes a file that was opened by the [LZOpenFile](#) or [OpenFile](#) function.

Parameter	Description
hf	Identifies the source file.

### Returns

This function does not return a value.

### Comments

If the file was compressed by Microsoft File Compression Utility (COMPRESS.EXE) and opened by the [LZOpenFile](#) function, LZClose frees any global heap space that was required to expand the file.

### Example

The following example uses LZClose to close a file opened by [LZOpenFile](#):

```
char szSrc[] = {"readme.txt"};
char szDst[] = {"readme.bak"};
OFSTRUCT ofStrSrc;
OFSTRUCT ofStrDest;
HFILE hfSrcFile, hfDstFile;

/* Open the source file. */

hfSrcFile = LZOpenFile(szSrc, &ofStrSrc, OF_READ);

/* Create the destination file. */

hfDstFile = LZOpenFile(szDst, &ofStrDest, OF_CREATE);

/* Copy the source file to the destination file. */

LZCopy(hfSrcFile, hfDstFile);

/* Close the files. */

LZClose(hfSrcFile);
LZClose(hfDstFile);

See Also
OpenFile, LZOpenFile
```

## LZCopy (3.1)

```
#include lzexpand.h
```

```
LONG LZCopy(hfSource, hfDest)
```

```
HFILE hfSource;          /* handle of source file      */
```

```
HFILE hfDest;           /* handle of destination      */
```

```
file
```

```
/
```

\*

The LZCopy function copies a source file to a destination file. If the source file was compressed by Microsoft File Compression Utility (COMPRESS.EXE), this function creates a decompressed destination file. If the source file was not compressed, this function duplicates the original file.

Parameter	Description
hfSource	Identifies the source file. (This handle is returned by the <a href="#">LZOpenFile</a> function when a compressed file is opened.)
hfDest	Identifies the destination file.

### Returns

The return value is the size, in bytes, of the destination file if the function is successful. Otherwise, it is an error value that is less than zero and may be one of the following:

Value	Meaning
<a href="#">LZERROR_BADINHANDLE</a>	The handle identifying the source file was not valid.
<a href="#">LZERROR_BADOUTHANDLE</a>	The handle identifying the destination file was not valid.
<a href="#">LZERROR_BADVALUE</a>	The input parameter is out of the allowable range.
<a href="#">LZERROR_GLOBALLOC</a>	There is insufficient memory for the required buffers.
<a href="#">LZERROR_GLOBLOCK</a>	The handle identifying the internal data structures is invalid.
<a href="#">LZERROR_READ</a>	The source file format was not valid.
<a href="#">LZERROR_UNKNOWNALG</a>	The source file was compressed with an unrecognized compression algorithm.
<a href="#">LZERROR_WRITE</a>	There is insufficient space for the output file.

### Comments

This function is identical to the [CopyLZFile](#) function.

If the function is successful, the file identified by hfDest is uncompressed.

If the source or destination file is opened by a C run-time function (rather than the [\\_lopen](#) or [OpenFile](#) function), it must be opened in binary mode.

### Example

The following example uses the LZCopy function to copy a file:

```
char szSrc[] = {"readme.txt"};
char szDst[] = {"readme.bak"};
OFSTRUCT ofStrSrc;
OFSTRUCT ofStrDest;
HFILE hfSrcFile, hfDstFile;

/* Open the source file. */

hfSrcFile = LZOpenFile(szSrc, &ofStrSrc, OF\_READ);

/* Create the destination file. */

hfDstFile = LZOpenFile(szDst, &ofStrDest, OF\_CREATE);

/* Copy the source file to the destination file. */

LZCopy(hfSrcFile, hfDstFile);

/* Close the files. */
```

**LZClose**(hfSrcFile);  
**LZClose**(hfDstFile);

See Also

CopyLZFile, \_lopen, LZOpenFile, OpenFile

LZERROR\_BADINHANDLE (-1)

The handle identifying the source file was not valid.

LZERROR\_BADINHANDLE (-1)

LZERROR\_BADOUTHANDLE (-2)

The handle identifying the destination file was not valid.



LZERROR\_BADOUTHANDLE (-2)

LZERROR\_BADVALUE (-7)

The input parameter is out of the allowable range.

LZERROR\_BADVALUE (-7)

LZERROR\_GLOBALLOC (-5)

There is insufficient memory for the required buffers.

LZERROR\_GLOBALLOC (-5)

LZERROR\_GLOBLOCK (-6)

The handle identifying the internal data structures is invalid.

LZERROR\_GLOBLOCK (-6)

LZERROR\_READ (-3)

The source file format was not valid.



LZERROR\_READ (-3)

LZERROR\_UNKNOWNALG (-8)

The source file was compressed with an unrecognized compression algorithm.

LZERROR\_UNKNOWNALG (-8)

LZERROR\_WRITE (-4)

There is insufficient space for the output file.

LZERROR\_WRITE (-4)

## LZDone (3.1)

```
#include lzexpand.h
void LZDone(void)
```

The LZDone function frees buffers that the LZStart function allocated for multiple-file copy operations.

Returns

This function does not return a value.

Comments

Applications that copy multiple files should call LZStart before copying the files with the CopyLZFile function. LZStart allocates buffers for the file copy operations.

Example

The following example uses LZDone to free buffers allocated by LZStart:

```
#define NUM_FILES 4

char *szSrc[NUM_FILES] =
    {"readme.txt", "data.txt", "update.txt", "list.txt"};
char *szDest[NUM_FILES] =
    {"readme.bak", "data.bak", "update.bak", "list.bak"};
OFSTRUCT ofStrSrc;
OFSTRUCT ofStrDest;
HFILE hfSrcFile, hfDstFile;
int i;

/* Allocate internal buffers for the CopyLZFile function. */

LZStart();

/* Open, copy, and then close the files. */

for (i = 0; i < NUM_FILES; i++) {
    hfSrcFile = LZOpenFile(szSrc[i], &ofStrSrc, OF_READ);
    hfDstFile = LZOpenFile(szDest[i], &ofStrDest, OF_CREATE);
    CopyLZFile(hfSrcFile, hfDstFile);
    LZClose(hfSrcFile);
    LZClose(hfDstFile);
}

LZDone(); /* free the internal buffers */
```

See Also

CopyLZFile, LZCopy, LZStart

## LZInit (3.1)

#include lzexpand.h

```
HFILE LZInit(hfSrc)
HFILE hfSrc; /* handle of source file */
```

The LZInit function allocates memory for, creates, and initializes the internal data structures that are required to decompress files.

Parameter	Description
hfSrc	Identifies the source file.

### Returns

The return value is the original file handle if the function is successful and the file is not compressed. If the function is successful and the file is compressed, the return value is a new file handle. If the function fails, the return value is an error value that is less than zero and may be one of the following:

Value	Meaning
LZERROR_BADINHANDLE	The handle identifying the source file is invalid.
LZERROR_GLOBALLOC	There is insufficient memory for the required internal data structures. This value is returned when an application attempts to open more than 16 files.
LZERROR_GLOBLOCK	The handle identifying global memory is invalid. (The internal call to the <u>GlobalLock</u> function failed.)
LZERROR_READ	The source file format is invalid.
LZERROR_UNKNOWNALG	The file was compressed with an unrecognized compression algorithm.

### Comments

A maximum of 16 compressed files can be open at any given time.

### Example

The following example uses LZInit to initialize the internal structures that are required to decompress a file:

```
char szSrc[] = {"readme.cmp"};
char szFileName[128];
OFSTRUCT ofStrSrc;
OFSTRUCT ofStrDest;
HFILE hfSrcFile, hfDstFile, hfCompFile;
int cbRead;
BYTE abBuf[512];

/* Open the compressed source file. */

hfSrcFile = OpenFile(szSrc, &ofStrSrc, OF_READ);

/*
 * Initialize internal data structures for the decompression
 * operation.
 */

hfCompFile = LZInit(hfSrcFile);

/* Retrieve the original name for the compressed file. */

GetExpandedName(szSrc, szFileName);

/* Create the destination file using the original name. */

hfDstFile = LZOpenFile(szFileName, &ofStrDest, OF_CREATE);

/* Copy the compressed source file to the destination file. */
```

```
do {
    if ((cbRead = LZRead(hfCompFile, abBuf, sizeof(abBuf))) > 0)
        lwrite(hfDstFile, abBuf, cbRead);
    else {
        . /* handle error condition */
        .
    }
} while (cbRead == sizeof(abBuf));

/* Close the files. */

LZClose(hfSrcFile);
LZClose(hfDstFile);
```



## LZOpenFile (3.1)

```
#include lzexpand.h
```

```
HFILE LZOpenFile(lpszFile, lpof, style)
LPCSTR lpszFile; /* address of filename */
OFSTRUCT FAR* lpof; /* address of structure for file info */
UINT style;
/* action to be taken */
```

The LZOpenFile function creates, opens, reopens, or deletes the file specified by the string to which lpszFile points.

Parameter	Description
lpszFile	Points to a string that specifies the name of a file.
lpof	Points to the <u>OFSTRUCT</u> structure that is to receive information about the file when the file is opened. The structure can be used in subsequent calls to LZOpenFile to refer to the open file. The szPathName member of this structure contains characters from the OEM character set.
style	Specifies the action to be taken. These styles can be combined by using the bitwise OR operator:

Value	Meaning
OF_CANCEL	Adds a Cancel button to the <u>OF_PROMPT</u> dialog box. Choosing the Cancel button directs LZOpenFile to return a file-not-found error message.
OF_CREATE	Directs LZOpenFile to create a new file. If the file already exists, it is truncated to zero length.
OF_DELETE	Deletes the file.
OF_EXIST	Opens the file, and then closes it. This action is used to test for file existence.
OF_PARSE	Fills the <u>OFSTRUCT</u> structure, but carries out no other action.
OF_PROMPT	Displays a dialog box if the requested file does not exist. The dialog box informs the user that Windows cannot find the file and prompts the user to insert the disk containing the file in drive A.
OF_READ	Opens the file for reading only.
OF_READWRITE	Opens the file for reading and writing.
OF_REOPEN	Opens the file using information in the reopen buffer.
OF_SHARE_DENY_NONE	Opens the file without denying other programs read access or write access to the file. LZOpenFile fails if the file has been opened in compatibility mode by any other program.
OF_SHARE_DENY_READ	Opens the file and denies other programs read access to the file. LZOpenFile fails if the file has been opened in compatibility mode or for read access by any other program.
OF_SHARE_DENY_WRITE	Opens the file and denies other programs write access to the file. LZOpenFile fails if the file has been opened in compatibility mode or for write access by any other program.
OF_SHARE_EXCLUSIVE	Opens the file in exclusive mode, denying other programs both read access and write access to the

file. LZOpenFile fails if the file has been opened in any other mode for read access or write access, even by the current program.  
Opens the file for writing only.

## OF\_WRITE

### Returns

The return value is a handle identifying the file if the function is successful and the value specified by style is not OF\_READ. If the file is compressed and opened with style set to the OF\_READ value, the return value is a special file handle. If the function fails, the return value is -1.

### Comments

If style is OF\_READ (or OF\_READ and any of the OF\_SHARE\_ flags) and the file is compressed, LZOpenFile calls the LZInit function, which performs the required initialization for the decompression operations.

### Example

The following example uses LZOpenFile to open a source file and create a destination file into which the source file can be copied:

```
char szSrc[] = {"readme.txt"};
char szDst[] = {"readme.bak"};
OFSTRUCT ofStrSrc;
OFSTRUCT ofStrDest;
HFILE hfSrcFile, hfDstFile;

/* Open the source file. */

hfSrcFile = LZOpenFile(szSrc, &ofStrSrc, OF_READ);

/* Create the destination file. */

hfDstFile = LZOpenFile(szDst, &ofStrDest, OF_CREATE);

/* Copy the source file to the destination file. */

LZCopy(hfSrcFile, hfDstFile);

/* Close the files. */

LZClose(hfSrcFile);
LZClose(hfDstFile);

See Also
LZInit
```

## LZRead (3.1)

#include lzexpand.h

```
int LZRead(hf, lpvBuf, cb)
HFILE hf;          /* handle of the file */
void FAR* lpvBuf; /* address of
buffer for file data
/
int cb;
* number of bytes to read
/
```

The LZRead function reads into a buffer bytes from a file.

Parameter	Description
hf	Identifies the source file.
lpvBuf	Points to a buffer that is to receive the bytes read from the file.
cb	Specifies the maximum number of bytes to be read.

### Returns

The return value is the actual number of bytes read if the function is successful. Otherwise, it is an error value that is less than zero and may be any of the following:

Value	Meaning
LZERROR_BADINHANDLE	The handle identifying the source file was invalid.
LZERROR_BADVALUE	The cb parameter specified a negative value.
LZERROR_GLOBLOCK	The handle identifying required initialization data is invalid.
LZERROR_READ	The format of the source file was invalid.
LZERROR_UNKNOWNALG	The file was compressed with an unrecognized compression algorithm.

### Comments

If the file is not compressed, LZRead calls the lread function, which performs the read operation.

If the file is compressed, LZRead emulates lread on an expanded image of the file and copies the bytes of data into the buffer to which lpvBuf points.

If the source file was compressed by Microsoft File Compression Utility (COMPRESS.EXE), the LZOpenFile, LZSeek, and LZRead functions can be called instead of the OpenFile, lseek, and lread functions.

### Example

The following example uses LZRead to copy and decompress a compressed file:

```
char szSrc[] = {"readme.cmp"};
char szFileName[128];
OFSTRUCT ofStrSrc;
OFSTRUCT ofStrDest;
HFILE hfSrcFile, hfDstFile, hfCompFile;
int cbRead;
BYTE abBuf[512];

/* Open the compressed source file. */

hfSrcFile = OpenFile(szSrc, &ofStrSrc, OF_READ);

/*
 * Initialize internal data structures for the decompression
 * operation.
 */

hfCompFile = LZInit(hfSrcFile);

/* Retrieve the original name for the compressed file. */
```

```
GetExpandedName(szSrc, szFileName);  
/* Create the destination file using the original name. */  
hfDstFile = LZOpenFile(szFileName, &ofStrDest, OF_CREATE);  
/* Copy the compressed source file to the destination file. */  
do {  
    if ((cbRead = LZRead(hfCompFile, abBuf, sizeof(abBuf))) > 0)  
        lwrite(hfDstFile, abBuf, cbRead);  
    else {  
        . /* handle error condition */  
        .  
    }  
} while (cbRead == sizeof(abBuf));  
/* Close the files. */  
LZClose(hfSrcFile);  
LZClose(hfDstFile);  
See Also  
_llseek, _lread, LZOpenFile, LZRead, LZSeek
```

### LZSeek (3.1)

```
#include lzexpand.h
```

```
LONG LZSeek(hf, lOffset, nOrigin)
```

```
HFILE hf; /* handle of file */
```

```
LONG lOffset; /* number of bytes to move */
```

```
/
```

```
int nOrigin;
```

```
* original position
```

```
/
```

The LZSeek function moves a file pointer from its original position to a new position.

Parameter	Description
hf	Identifies the source file.
lOffset	Specifies the number of bytes by which the file pointer should be moved.
nOrigin	Specifies the starting position of the pointer. This parameter must be one of the following values:

Value	Meaning
0	Move the file pointer lOffset bytes from the beginning of the file.
1	Move the file pointer lOffset bytes from the current position.
2	Move the file pointer lOffset bytes from the end of the file.

#### Returns

The return value is the offset from the beginning of the file to the new pointer position, if the function is successful. Otherwise, it is an error value that is less than zero and may be one of the following:

Value	Meaning
LZERROR_BADINHANDLE	The handle identifying the source file was invalid.
LZERROR_BADVALUE	One of the parameters exceeds the range of valid values.
LZERROR_GLOBLOCK	The handle identifying the initialization data is invalid.

#### Comments

If the file is not compressed, LZSeek calls the llseek function and moves the file pointer by the specified offset.

If the file is compressed, LZSeek emulates llseek on an expanded image of the file.

See Also

llseek

## LZStart (3.1)

```
#include lzexpand.h  
int LZStart(void)
```

The LZStart function allocates the buffers that the CopyLZFile function uses to copy a source file to a destination file.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is LZERROR\_GLOBALLOC.

### Comments

Applications that copy (or copy and decompress) multiple consecutive files should call the LZStart, CopyLZFile, and LZDone functions. Applications that copy a single file should call the LZCopy function.

### Example

The following example uses LZStart to allocate buffers used by CopyLZFile:

```
#define NUM_FILES    4  
  
char *szSrc[NUM_FILES] =  
    {"readme.txt", "data.txt", "update.txt", "list.txt"};  
char *szDest[NUM_FILES] =  
    {"readme.bak", "data.bak", "update.bak", "list.bak"};  
OFSTRUCT ofStrSrc;  
OFSTRUCT ofStrDest;  
HFILE hfSrcFile, hfDstFile;  
int i;  
  
/* Allocate internal buffers for the CopyLZFile function. */  
  
LZStart();  
  
/* Open, copy, and then close the files. */  
  
for (i = 0; i < NUM_FILES; i++) {  
    hfSrcFile = LZOpenFile(szSrc[i], &ofStrSrc, OF_READ);  
    hfDstFile = LZOpenFile(szDest[i], &ofStrDest, OF_CREATE);  
    CopyLZFile(hfSrcFile, hfDstFile);  
    LZClose(hfSrcFile);  
    LZClose(hfDstFile);  
}  
  
LZDone(); /* free the internal buffers */  
  
See Also  
CopyLZFile, LZCopy, LZDone
```

## Lempel-Ziv Encoding functions

<u>CopyLZFile</u>	Copies files and decompresses them if compressed
<u>GetExpandedName</u>	Retrieves the original filename of a compressed file
<u>LZClose</u>	Closes a file
<u>LZCopy</u>	Copies a file and decompresses it if compressed
<u>LZDone</u>	Frees buffers allocated by the LZStart function
<u>LZInit</u>	Initializes structures needed for decompression
<u>LZOpenFile</u>	Opens a file (both compressed and uncompressed)
<u>LZRead</u>	Reads a specified number of bytes from a compressed file
<u>LZSeek</u>	Repositions a pointer in a file
<u>LZStart</u>	Allocates buffers for the CopyLZFile function

## DECLARE\_HANDLE (3.1)

DECLARE\_HANDLE(name)

The DECLARE\_HANDLE macro creates a data type that can be used to define 16-bit handles.

Parameter	Description
name	Specifies the name of the new data type.

### Comments

The DECLARE\_HANDLE macro is defined in WINDOWS.H as follows:

```
#define DECLARE_HANDLE(name) struct name##_ { int unused; }; \  
    typedef const struct name##_ NEAR* name
```

### See Also

[DECLARE\\_HANDLE32](#)



## DECLARE\_HANDLE32 (3.1)

#include ddeml.h

DECLARE\_HANDLE32(name)

The DECLARE\_HANDLE32 macro creates a data type that can be used to define 32-bit handles.

Parameter	Description
name	Specifies the name of the new data type.
Parameter	Description
name	Specifies the name of the variable for which a pointer is created.

### Comments

The DECLARE\_HANDLE32 macro is defined in DDEML.H as follows:

```
#define DECLARE_HANDLE32(name) struct name##_ { int unused; }; \  
typedef const struct name##_ _far* name
```

### See Also

[DECLARE\\_HANDLE](#)

## FIELDOFFSET (3.1)

int FIELDOFFSET(type, field)

The FIELDOFFSET macro computes the address offset of the specified member in the structure specified by the type parameter.

Parameter	Description
type	Specifies the name of the structure.
field	Specifies the name of the member defined within the given structure.

### Returns

The return value is the address offset of the given structure member.

### Comments

The FIELDOFFSET macro is defined in WINDOWS.H as follows:

```
#define FIELDOFFSET(type, field) ((int)(amp((type NEAR*)1)->field)-1)
```

## GetBValue (3.1)

```
BYTE GetBValue(rgb)
DWORD rgb; /* RGB color value */
```

The GetBValue macro extracts the intensity value of the blue color field from the 32-bit integer value specified by the rgb parameter.

Parameter	Description
rgb	Specifies the <u>RGB</u> color value.

### Returns

The return value specifies the intensity of the blue color field.

### Comments

The GetBValue macro is defined in WINDOWS.H as follows:

```
#define GetBValue(rgb) ((BYTE)((rgb)>>16))
```

### See Also

GetGValue, GetRValue, RGB

## GetGValue (3.1)

```
BYTE GetGValue(rgb)
DWORD rgb;    /* RGB color value */
```

The GetGValue macro extracts the intensity value of the green color field from the 32-bit integer value specified by the rgb parameter.

Parameter	Description
rgb	Specifies the <u>RGB</u> color value.

### Returns

The return value specifies the intensity of the green color field.

### Comments

The GetGValue macro is defined in WINDOWS.H as follows:

```
#define GetGValue(rgb) ((BYTE)(((WORD)(rgb)) >> 8))
```

### See Also

GetBValue, GetRValue, RGB

## GetRValue (3.1)

```
BYTE GetRValue(rgb)
DWORD rgb;    /* RGB color value */
```

The GetRValue macro extracts the intensity value of the red color field from the 32-bit integer value specified by the rgb parameter.

Parameter	Description
rgb	Specifies the <u>RGB</u> color value.

### Returns

The return value specifies the intensity of the red color field.

### Comments

The GetRValue macro is defined in WINDOWS.H as follows:

```
#define GetRValue(rgb) ((BYTE)(rgb))
```

### See Also

GetBValue, GetGValue, RGB

## GlobalDiscard (2.x)

```
HGLOBAL GlobalDiscard(hglb)
HGLOBAL hglb;      /* handle of object to discard */
```

The GlobalDiscard macro discards the given global memory object. The lock count of the memory object must be zero.

Parameter	Description
hglb	Identifies the global memory object to be discarded.

### Returns

The return value is a handle of the discarded object if the macro is successful. Otherwise, it is NULL.

### Comments

The GlobalDiscard macro discards only global objects that an application allocated with the GMEM\_DISCARDABLE and GMEM\_MOVEABLE flags set. The macro fails if an application attempts to discard a fixed or locked object.

Although GlobalDiscard removes the global memory object from memory, the object's handle remains valid. An application can subsequently pass the handle to the GlobalReAlloc function to allocate another global memory object identified by the same handle.

The GlobalDiscard macro is defined in WINDOWS.H as follows:

```
#define GlobalDiscard(h) GlobalReAlloc(h, 0L, GMEM_MOVEABLE)
```

### See Also

GlobalFree, GlobalReAlloc

## HIBYTE (2.x)

```
BYTE HIBYTE(wInteger)
UINT uVal;      /* value from which high byte is retrieved */
```

The HIBYTE macro retrieves the high-order byte from the integer value specified by the wInteger parameter.

Parameter	Description
uVal	Specifies the value to be converted.

**Returns**  
The return value specifies the high-order byte of the given value.

**Comments**  
The HIBYTE macro is defined in WINDOWS.H as follows:

```
#define HIBYTE(w) ((BYTE)(((UINT)(w) >> 8) & 0xFF))
```

## HIWORD (2.x)

WORD HIWORD(dwInteger)

DWORD dwInteger; /\* value from which high word is retrieved \*/

The HIWORD macro retrieves the high-order word from the 32-bit integer value specified by the dwInteger parameter.

Parameter	Description
dwInteger	Specifies the value to be converted.

### Returns

The return value specifies the high-order word of the given 32-bit integer value.

### Comments

The HIWORD macro is defined in WINDOWS.H as follows:

```
#define HIWORD(l) ((WORD)((((DWORD)(l)) >> 16) & 0xFFFF))
```



## LOBYTE (2.x)

```
BYTE LOBYTE(uVal)
UINT uVal;      /* value from which low byte is retrieved */
```

The LOBYTE macro extracts the low-order byte from the 16-bit integer value specified by the uVal parameter.

Parameter	Description
uVal	Specifies the value to be converted.

**Returns**  
The return value specifies the low-order byte of the value.

**Comments**  
The LOBYTE macro is defined in WINDOWS.H as follows:

```
#define LOBYTE(w) ((BYTE)(w))
```

See Also  
[LOWORD](#)

## LocalDiscard (2.x)

```
HLOCAL LocalDiscard(hloc)
HLOCAL hloc;      /* handle of object to discard */
```

The LocalDiscard macro discards the given local memory object. The lock count of the memory object must be zero.

Parameter	Description
hloc	Identifies the local memory object to be discarded.

### Returns

The return value is equal to the hloc parameter if the macro is successful. Otherwise, it is NULL.

### Comments

Although the LocalDiscard macro removes the local memory object from memory, the object's handle remains valid. An application can subsequently pass the handle to the LocalReAlloc function to allocate another local memory object identified by the same handle.

The LocalLock function increments (increases by one) a memory object's lock count. The LocalUnlock function decrements (decreases by one) the lock count.

The LocalDiscard macro is defined in WINDOWS.H as follows:

```
#define LocalDiscard(h) LocalReAlloc(h, 0, LMEM_MOVEABLE)
```

### See Also

LocalLock, LocalReAlloc, LocalUnlock

## LockData (2.x)

HANDLE LockData(dummy)

The LockData macro locks the current data segment in memory. It is intended to be used in modules that have movable data segments.

Parameter	Description
dummy	This parameter is ignored.

### Returns

The return value identifies the locked data segment if the function is successful. Otherwise, it is NULL.

### Comments

The LockData macro is defined in WINDOWS.H as follows:

```
#define LockData(dummy) LockSegment((UINT)-1)
```

### See Also

LockSegment

## LOWORD (2.x)

WORD LOWORD(dwVal)

DWORD dwVal; /\* value from which low word is retrieved \*/

The LOWORD macro extracts the low-order word from the 32-bit integer value specified by the dwVal parameter.

Parameter	Description
dwVal	Specifies the value to be converted.

### Returns

The return value specifies the low-order word of the 32-bit integer value.

### Comments

The LOWORD macro is defined in WINDOWS.H as follows:

```
#define LOWORD(l) ((WORD)(DWORD)(l))
```

See Also

LOBYTE

## MAKEINTATOM (2.x)

```
LPCSTR MAKEINTATOM(wInteger)
WORD wInteger;      /* integer to make into atom */
```

The MAKEINTATOM macro creates an integer atom that represents a character string of decimal digits. Integer atoms created by this macro can be added to the atom table using the [AddAtom](#) function.

Parameter	Description
wInteger	Specifies the numeric value to be made into an integer atom.

### Returns

The return value is a pointer to the atom created for the given integer.

### Comments

Although the return value of the MAKEINTATOM macro is cast as an [LPCSTR](#), the return value cannot be used as a string pointer, except when it is passed to atom-management functions that require an [LPCSTR](#) parameter.

The [DeleteAtom](#) function always succeeds for integer atoms, even though it does nothing. The string returned by the [GetAtomName](#) function for an integer atom will be a null-terminated string where the first character is a pound sign (#) and the remaining characters are the word used in the MAKEINTATOM macro.

The MAKEINTATOM macro is defined in WINDOWS.H as follows:

```
#define MAKEINTATOM(i) ((LPCSTR)MAKELP(NULL, (i)))
```

### Example

The following example uses the MAKEINTATOM macro to convert the number 32,565 into an integer atom. The atom is then added to the local atom table by the [AddAtom](#) function:

```
ATOM at;
char szMsg[80];
LPCSTR lpszAtom;

lpszAtom = MAKEINTATOM(32565);
at = AddAtom(lpszAtom);

if (at == 0)
    MessageBox(hwnd, "AddAtom failed", "", MB_ICONSTOP);
else {
    wsprintf(szMsg, "AddAtom returned %u", at);
    MessageBox(hwnd, szMsg, "", MB_OK);
}
```

### See Also

[AddAtom](#), [DeleteAtom](#), [GetAtomName](#)

## MAKEINTRESOURCE (2.x)

```
LPCSTR MAKEINTRESOURCE(idResource)  
WORD idResource;      /* resource identifier to convert */
```

The MAKEINTRESOURCE macro converts an integer resource identifier into a value compatible with Windows resource-management functions. This macro is used in place of a string containing the name of the resource.

Parameter	Description
idResource	Specifies the integer resource identifier to be converted.

### Returns

The return value contains the idResource parameter in the low-order word and zero in the high-order word.

### Comments

The MAKEINTRESOURCE macro is defined in WINDOWS.H as follows:

```
#define MAKEINTRESOURCE(i) ((LPCSTR)MAKELP(NULL, (i)))
```

### See Also

MAKELP

## MAKELONG (2.x)

```
DWORD MAKELONG(uLow, uHigh)
UINT uLow;          /* low-order word of long value */
UINT uHigh;         /* high-order word of long value */
long value
/
```

The MAKELONG macro creates an unsigned long integer by concatenating two integer values, specified by the uLow and uHigh parameters.

Parameter	Description
uLow	Specifies the low-order word of the new long value.
uHigh	Specifies the high-order word of the new long value.

### Returns

The return value specifies an unsigned long-integer value.

### Comments

The MAKELONG macro is defined in WINDOWS.H as follows:

```
#define MAKELONG(low, high) \
    ((LONG)((WORD)(low) | (((DWORD)((WORD)(high))) << 16)))
```

## MAKELP (3.1)

```
void FAR* MAKELP(wSel, wOff)
WORD wSel;    /* selector */
WORD wOff;    /* offset */
```

The MAKELP macro combines a segment selector and an address offset to create a long (32-bit) pointer to a memory address.

Parameter	Description
wSel	Specifies a segment selector.
wOff	Specifies an offset from the beginning of the given segment to the desired byte.

### Returns

The return value is a long pointer to an unspecified data type.

### Comments

The MAKELP macro is defined in WINDOWS.H as follows:

```
#define MAKELP(sel, off) ((void FAR*)MAKELONG((off), (sel)))
```

### See Also

MAKELONG



## MAKELPARAM (3.1)

LPARAM MAKELPARAM(wLow, wHigh)

WORD wLow; /\* low-order word \*/

WORD wHigh; /\* high-order word \*/

The MAKELPARAM macro creates an unsigned long integer for use as an lParam parameter in a message. The macro concatenates two integer values, specified by the wLow and wHigh parameters.

Parameter	Description
wLow	Specifies the low-order word of the new long value.
wHigh	Specifies the high-order word of the new long value.

### Returns

The return value specifies an unsigned long-integer value.

### Comments

The MAKELPARAM macro is defined in WINDOWS.H as follows:

```
#define MAKELPARAM(low, high) ((LPARAM)MAKELONG(low, high))
```

### See Also

MAKELONG, MAKELRESULT

## MAKELRESULT (3.1)

LRESULT MAKELRESULT(wLow, wHigh)

WORD wLow; /\* low-order word \*/

WORD wHigh; /\* high-order word \*/

The MAKELRESULT macro creates an unsigned long integer for use as a return value from a window procedure. The macro concatenates two integer values, specified by the wLow and wHigh parameters.

Parameter	Description
wLow	Specifies the low-order word of the new long value.
wHigh	Specifies the high-order word of the new long value.

### Returns

The return value specifies an unsigned long-integer value.

### Comments

The MAKELRESULT macro is defined in WINDOWS.H as follows:

```
#define MAKELRESULT(low, high) ((LRESULT)MAKELONG(low, high))
```

### See Also

MAKELONG, MAKELPARAM

## MAKEPOINT (2.x)

```
POINT MAKEPOINT(lval)
DWORD lval;    /* coordinates of a point */
```

The MAKEPOINT macro converts a long value that contains the x- and y-coordinates of a point into a POINT structure. This macro is useful for converting the long value returned by the GetMessagePos function into a POINT structure and for converting the lParam value passed with mouse messages into a POINT structure containing the mouse coordinates.

Parameter	Description
lval	Specifies the coordinates of a point. The x-coordinate is in the low-order word, and the y-coordinate is in the high-order word.

### Returns

The return value is a pointer to a POINT structure.

### Comments

The MAKEPOINT macro is defined in WINDOWS.H as follows:

```
#define MAKEPOINT(l)  (*( (POINT FAR*)&(l) ))
```

The MAKEPOINT macro is not compatible with the Windows 32-bit application programming interface (API).

### See Also

GetMessagePos, POINT

max (2.x)

int max(value1, value2)

The max macro compares two values and returns the value of the larger one. The data type can be any numerical data type, signed or unsigned. The type of the arguments and the return value is the same.

Parameter	Description
value1	Specifies the first of two values.
value2	Specifies the second of two values.

Returns

The return value is value1 or value2, whichever is greater.

Comments

The max macro is defined in WINDOWS.H as follows:

```
#define max(a, b) (((a) > (b)) ? (a) : (b))
```

See Also

min

**min (2.x)**

`int min(value1, value2)`

The min macro compares two values and returns the value of the smaller one. The data type can be any numerical data type, signed or unsigned. The type of the arguments and the return value is the same.

Parameter	Description
value1	Specifies the first of two values.
value2	Specifies the second of two values.

**Returns**

The return value is value1 or value2, whichever is smaller.

**Comments**

The min macro is defined in WINDOWS.H as follows:

```
#define min(a, b) ((a) < (b)) ? (a) : (b)
```

**See Also**

max

## OFFSETOF (3.1)

```
WORD OFFSETOF(lp)
void FAR* lp;      /* long pointer */
```

The OFFSETOF macro retrieves the address offset of the specified long pointer.

Parameter	Description
lp	Specifies a long pointer.

### Returns

The return value is the offset address.

### Comments

The OFFSETOF macro is defined in WINDOWS.H as follows:

```
#define OFFSETOF(lp)    LOWORD(lp)
```

### See Also

LOWORD, SELECTOROF

## PALETTEINDEX (3.0)

```
COLORREF PALETTEINDEX(wPaletteIndex)
WORD wPaletteIndex; /* index to palette entry */
```

The PALETTEINDEX macro accepts an index to a logical-color palette entry and returns a value consisting of 1 in the high-order byte and the palette-entry index in the low-order byte. This is called a palette-entry specifier. An application using a color palette can pass this specifier instead of an explicit RGB value to functions that expect a color. This allows the function to use the color in the specified palette entry.

Parameter	Description
wPaletteIndex	Specifies an index to the palette entry containing the color to be used for a graphics operation.

### Returns

The return value is a logical-palette index specifier. When using a logical palette, an application can use this specifier in place of an explicit RGB value for graphics-device interface (GDI) functions that require a color.

### Comments

The PALETTEINDEX macro is defined in WINDOWS.H as follows:

```
#define PALETTEINDEX(i) ((COLORREF)(0x01000000L | (DWORD)(WORD)(i)))
```

### See Also

PALETTERGB, RGB

## PALETTERGB (3.0)

```
COLORREF PALETTERGB(cRed, cGreen, cBlue)
BYTE cRed;          /* red component of palette-relative RGB */
BYTE cGreen;        /* green
component of palette-relative RGB
/
BYTE cBlue;
* blue component of palette-relative RGB
/
```

The PALETTERGB macro accepts three values representing relative intensities of red, green, and blue and returns a value consisting of 2 in the high-order byte and an RGB value in the three low-order bytes. This is called a palette-relative RGB specifier. An application using a color palette can pass this specifier instead of an explicit RGB value to functions that expect a color.

For output devices that support logical palettes, Windows matches a palette-relative RGB value to the nearest color in the logical palette of the device context as though the application had specified an index to that palette entry. If an output device does not support a system palette, then Windows uses the palette-relative RGB as though it were a conventional RGB doubleword returned by the RGB macro.

Parameter	Description
cRed	Specifies the intensity of the red color field.
cGreen	Specifies the intensity of the green color field.
cBlue	Specifies the intensity of the blue color field.

### Returns

The return value specifies a palette-relative RGB value.

### Comments

The PALETTERGB macro is defined in WINDOWS.H as follows:

```
#define PALETTERGB(r,g,b)    (0x02000000L | RGB(r,g,b))
```

### See Also

PALETTEINDEX, RGB



## RGB (2.x)

```
COLORREF RGB(cRed, cGreen, cBlue)
BYTE cRed;          /* red component of color */
BYTE cGreen;        /* green component of color */
/
BYTE cBlue;
* blue component of color
/
```

The RGB macro selects an RGB color based on the parameters supplied and the color capabilities of the output device.

Parameter	Description
cRed	Specifies the intensity of the red color field.
cGreen	Specifies the intensity of the green color field.
cBlue	Specifies the intensity of the blue color field.

### Returns

The return value specifies the resultant RGB color.

### Comments

The intensity for each argument can range from 0 through 255. If all three intensities are specified as zero, the result is black. If all three intensities are specified as 255, the result is white.

### Comments

The RGB macro is defined in WINDOWS.H as follows:

```
#define RGB(r,g,b) ((COLORREF)(((BYTE)(r)|((WORD)(g)<<8))| \
((DWORD)(BYTE)(b))<<16))
```

### See Also

[GetBValue](#), [GetGValue](#), [GetRValue](#), [PALETTEINDEX](#), [PALETTERGB](#)

## SELECTOROF (3.1)

WORD SELECTOROF(lp)

```
void FAR* lp;      /* long pointer */
```

The SELECTOROF macro retrieves the segment selector from the specified long pointer.

Parameter	Description
lp	Specifies a long pointer.

Returns

The return value is the segment selector.

Comments

The SELECTOROF macro is defined in WINDOWS.H as follows:

```
#define SELECTOROF(lp) HIWORD(lp)
```

See Also

HIWORD, OFFSETOF

## UnlockData (2.x)

HANDLE UnlockData(dummy)

The UnlockData macro unlocks the current data segment. It is intended to be used by modules that have movable data segments.

Parameter	Description
dummy	This parameter is ignored.

### Returns

The return value specifies the outcome of the UnlockSegment function. It is zero if the segment's lock count was decreased to zero. Otherwise, the return value is nonzero.

### Comments

The UnlockData macro is defined in WINDOWS.H as follows:

```
#define UnlockData(dummy)    UnlockSegment((UINT)-1)
```

### See Also

LockData, UnlockSegment

## UnlockResource (2.x)

```
BOOL UnlockResource(hgblResData)
HGLOBAL hgblResData;          /* handle of memory object to unlock */
```

The `UnlockResource` macro unlocks the resource specified by the `hgblResData` parameter and decreases the reference count of the resource by one.

Parameter	Description
<code>hgblResData</code>	Identifies the global memory object to be unlocked.

### Returns

The return value is zero if the object's reference count is decreased to zero. Otherwise, it is nonzero.

### Comments

The `UnlockResource` macro is defined in `WINDOWS.H` as follows:

```
#define UnlockResource(h)    GlobalUnlock(h)
```

### See Also

[GlobalUnlock](#)

### Windows macros (3.1)

<u>DECLARE_HANDLE</u>	Creates a 32-bit handle data type
<u>DECLARE_HANDLE32</u>	Creates a 16-bit handle data type
<u>FIELDOFFSET</u>	Computes the address offset of a structure member
<u>GetBValue</u>	Retrieves the blue color field of an <u>RGB</u> value
<u>GetGValue</u>	Retrieves the green color field of an <u>RGB</u> value
<u>GetRValue</u>	Retrieves the red color field of an <u>RGB</u> value
<u>GlobalDiscard</u>	Discards a global memory object
<u>HIBYTE</u>	Retrieves the high-order byte of an integer
<u>HIWORD</u>	Retrieves the high-order word of a 32-bit integer
<u>LOBYTE</u>	Retrieves the low-order byte of a 16-bit integer
<u>LocalDiscard</u>	Discards a local memory object
<u>LockData</u>	Locks the current data segment in memory
<u>LOWORD</u>	Retrieves the low-order word of a 32-bit integer
<u>MAKEINTATOM</u>	Creates an integer atom
<u>MAKEINTRESOURCE</u>	Converts a resource identifier into a string
<u>MAKELONG</u>	Creates a 32-bit integer from two 16-bit integers
<u>MAKELP</u>	Creates a long pointer to a memory address
<u>MAKELPARAM</u>	Creates unsigned long-integer message parameter
<u>MAKELRESULT</u>	Creates unsigned long-integer message result
<u>MAKEPOINT</u>	Converts a long value into a <u>POINT</u> structure
<u>max</u>	Returns the larger of two values
<u>min</u>	Returns the smaller of two values
<u>OFFSETOF</u>	Retrieves the address offset of a long pointer
<u>PALETTEINDEX</u>	Returns a logical-palette index specifier
<u>PALETTERGB</u>	Returns a palette-relative <u>RGB</u> specifier
<u>RGB</u>	Creates an <u>RGB</u> value from <u>three</u> colors
<u>SELECTOROF</u>	Retrieves the segment selector from a long pointer
<u>UnlockData</u>	Unlocks the current data segment
<u>UnlockResource</u>	Unlocks a resource and decreases its reference count





Information in this document is subject to change without notice and does not represent a commitment on the part of Microsoft Corporation. The software, which includes information contained in any databases, described in this document is furnished under a license agreement or nondisclosure agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the license or nondisclosure agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the express written permission of Microsoft Corporation.

(C) Copyright Microsoft Corporation, 1992. All rights reserved.  
Simultaneously published in the U.S. and Canada.



## MS Windows (3.1)

Copyright

Database version

New 3.1 API

Alphabetical functions

Function Groups

Macros

Messages

Notification messages

Overviews

Printer escapes

Resource statements

Structures

Tables

Tools

New 3.1 API

Common dialog box functions

DDE functions

Toolhelp functions

Drag-drop functions

Installable-driver functions

Lempel-Ziv Encoding functions

OLE functions

Printer escapes

Registration functions

Shell functions

Stress functions

TrueType functions

Version functions

New 3.1 messages

New 3.1 GDI functions

New 3.1 kernel functions

New 3.1 user functions

Microsoft Windows  
Software Development Kit

Version 3.1

## Windows function groups (3.1)

32-bit Memory Management functions

Application-Execution functions

Atom functions

Bitmap functions

Brush functions

Callback functions

Caret functions

Clipboard functions

Clipping functions

Common-dialog box functions

Communication functions

Coordinate functions

Cursor functions

DDE functions

Debugging functions

Device-Context functions

Dialog-Box functions

Display and movement functions

Drag-drop functions

Drawing-Attribute functions

Drawing-Tool functions

Ellipse and Polygon functions

Error functions

File I/O functions

Font functions

GDI functions

Hardware functions

Hook functions

Icon functions

Information functions

Initialization-File functions

Input functions

Installable-Driver functions

Kernel functions

Lempel-Ziv Encoding functions

Line-Output functions

Mapping functions

Memory-Management functions

Menu functions

Message functions

Metafile functions

Module-Management functions

OLE function Groups

Optimization-Tool functions

Painting functions

Palette functions

Pen functions

Pointer Validation Functions

Printer-Control functions

Property functions

Rectangle functions

Region functions

Registration functions

Resource-Management functions

Screen-Saver functions

Scrolling functions

Segment functions

Shell functions

Stress functions

String-Manipulation functions

System functions

Task functions

Text functions

Toolhelp functions

TrueType functions

User functions

Version functions

Window-Creation functions

### 32-bit Memory Manager Functions

<u>GetWinMem32Version</u>	Retrieves the version of the 32-bit memory API
<u>Global16PointerAlloc</u>	Converts a 16:32 pointer to a 16:16 pointer alias
<u>Global16PointerFree</u>	Frees a 16:16 pointer alias
<u>Global32Alloc</u>	Allocates a USE32 memory object
<u>Global32CodeAlias</u>	Creates a USE32 alias selector for 32-bit object
<u>Global32CodeAliasFree</u>	Frees a USE32 code-segment alias selector
<u>Global32Free</u>	Frees a USE32 memory object
<u>Global32Realloc</u>	Changes the size of a USE32 memory object

Alphabetical functions

Callback functions

Common dialog box functions

DDEML functions

Drag-drop functions

GDI functions

Kernel functions

LZExpand functions

OLE functions

Registration functions

Screen saver functions

Shell functions

Stress functions

Toolhelp functions

User functions

Version functions

### Caret Functions (3.1)

<u>CreateCaret</u>	Creates a new shape for the system caret
<u>DestroyCaret</u>	Destroys the current caret shape
<u>GetCaretBlinkTime</u>	Retrieves the caret blink rate
<u>GetCaretPos</u>	Retrieves the current caret position
<u>HideCaret</u>	Removes the caret from the screen
<u>SetCaretBlinkTime</u>	Sets the caret blink rate
<u>SetCaretPos</u>	Sets the caret position
<u>ShowCaret</u>	Shows (unhides) the caret on the screen

## Clipboard Functions (3.1)

The clipboard provides a mechanism that makes it possible for applications to pass data handles to other applications. Clipboard functions carry out data interchange between Windows applications.

<u>ChangeClipboardChain</u>	Removes window from clipboard-viewer chain
<u>CloseClipboard</u>	Closes the clipboard
<u>CountClipboardFormats</u>	Retrieves the number of clipboard formats
<u>EmptyClipboard</u>	Empties the clipboard and frees data handles
<u>EnumClipboardFormats</u>	Returns the available clipboard formats
<u>GetClipboardData</u>	Retrieves a handle to clipboard data
<u>GetClipboardFormatName</u>	Retrieves the registered clipboard-format name
<u>GetClipboardOwner</u>	Retrieves clipboard-owner window handle
<u>GetClipboardViewer</u>	Retrieves first clipboard-viewer window handle
<u>GetPriorityClipboardFormat</u>	Retrieves first clipboard format in priority list
<u>IsClipboardFormatAvailable</u>	Determines whether format data is available
<u>OpenClipboard</u>	Opens the clipboard
<u>RegisterClipboardFormat</u>	Registers a new clipboard format
<u>SetClipboardData</u>	Sets the data in the clipboard
<u>SetClipboardViewer</u>	Adds a window to the clipboard-viewer chain



### Cursor Functions (3.1)

<u>ClipCursor</u>	Confines the cursor to a specified rectangle
<u>CreateCursor</u>	Creates a cursor with specified dimensions
<u>DestroyCursor</u>	Destroys a cursor created by CreateCursor or LoadCursor
<u>GetCursorPos</u>	Retrieves the current cursor position in screen coordinates
<u>LoadCursor</u>	Loads a cursor resource
<u>SetCursor</u>	Changes the mouse cursor
<u>SetCursorPos</u>	Sets the mouse-cursor position in screen coordinates
<u>ShowCursor</u>	Shows or hides the mouse cursor

### Dialog-Box Functions (3.1)

<u>CheckDlgButton</u>	Changes a check mark by a dialog box button
<u>CheckRadioButton</u>	Adds a check mark to a radio button
<u>CreateDialog</u>	Creates a modeless dialog box
<u>CreateDialogIndirect</u>	Creates modeless dialog box from memory template
<u>CreateDialogIndirectParam</u>	Creates modeless dialog box from memory template
<u>CreateDialogParam</u>	Creates a modeless dialog box
<u>DefDlgProc</u>	Provides default window message processing
<u>DialogBox</u>	Creates a modal dialog box
<u>DialogBoxIndirect</u>	Creates modal dialog box from memory template
<u>DialogBoxIndirectParam</u>	Creates modal dialog box from memory template
<u>DialogBoxParam</u>	Creates a modal dialog box
<u>DlgDirList</u>	Fills a directory list box
<u>DlgDirListComboBox</u>	Fills a directory list box
<u>DlgDirSelect</u>	Retrieves a selection from a directory list box
<u>DlgDirSelectEx</u>	Retrieves a selection from a directory list box
<u>DlgDirSelectComboBox</u>	Retrieves a selection from a directory list box
<u>DlgDirSelectComboBoxEx</u>	Retrieves a selection from a directory list box
<u>EndDialog</u>	Hides a modal dialog box
<u>GetDialogBaseUnits</u>	Returns dialog base units
<u>GetDlgCtrlID</u>	Returns the handle of a child window
<u>GetDlgItem</u>	Retrieves the handle of a dialog box control
<u>GetDlgItemInt</u>	Translates control text into an integer
<u>GetDlgItemText</u>	Retrieves control text or title
<u>GetNextDlgGroupItem</u>	Returns handle of previous or next group control
<u>GetNextDlgTabItem</u>	Returns the next or previous <code>WS_TABSTOP</code> control
<u>IsDialogMessage</u>	Determines if message is for modeless dialog box
<u>IsDlgButtonChecked</u>	Determines the state of a button
<u>MapDialogRect</u>	Maps dialog box units to pixels
<u>SendDlgItemMessage</u>	Sends a message to a dialog box control
<u>SetDlgItemInt</u>	Converts an integer to a dialog box string
<u>SetDlgItemText</u>	Sets title or text of a control

### Display and Movement Functions (3.1)

<u>ArrangeIconicWindows</u>	Arranges minimized child windows
<u>BeginDeferWindowPos</u>	Creates a window-position structure
<u>BringWindowToTop</u>	Uncovers an overlapped window
<u>CloseWindow</u>	Minimizes (but does not destroy) a window
<u>DeferWindowPos</u>	Updates a window-position structure
<u>EndDeferWindowPos</u>	Updates the position and size of multiple windows
<u>GetClientRect</u>	Retrieves a window's client coordinates
<u>GetWindowRect</u>	Retrieves a window's screen coordinates
<u>GetWindowText</u>	Copies window title or control text to a buffer
<u>GetWindowTextLength</u>	Returns length of window title or control text
<u>IsIconic</u>	Determines whether a window is minimized
<u>IsWindowVisible</u>	Determines the visibility state of a window
<u>IsZoomed</u>	Determines whether a window is maximized
<u>MoveWindow</u>	Changes the position and dimensions of a window
<u>OpenIcon</u>	Activates and displays a minimized window
<u>SetWindowPos</u>	Sets a window's size, position, and z-order
<u>SetWindowText</u>	Sets control text or window title
<u>ShowOwnedPopups</u>	Shows or hides pop-up windows
<u>ShowWindow</u>	Sets a window's visibility state

### Error Functions (3.1)

<u>FlashWindow</u>	Flashes a window once
<u>MessageBeep</u>	Generates a beep sound
<u>MessageBox</u>	Creates and displays a message-box window

### Hardware Functions (3.1)

<u>EnableHardwareInput</u>	Controls mouse and keyboard input queuing
<u>GetAsyncKeyState</u>	Determines the key state
<u>GetInputState</u>	Determines mouse, keyboard and timer queuing status
<u>GetKeyboardState</u>	Copies virtual-keyboard keys status to a buffer
<u>GetKeyNameText</u>	Retrieves the string representing the name of a key
<u>GetKeyState</u>	Retrieves the virtual-key state
<u>GetKBCodePage</u>	Returns the current Windows code page
<u>OemKeyScan</u>	Translates OEM ASCII to scan codes
<u>SetKeyboardState</u>	Sets the Windows keyboard-state table
<u>MapVirtualKey</u>	Translates a virtual-key code or scan code
<u>VkKeyScan</u>	Translates a Windows character to a virtual-key code

### Hook Functions (3.1)

<u>CallMsgFilter</u>	Passes a message to a message-filter function
<u>DefHookProc</u>	Calls the next function in a hook-function chain
<u>SetWindowsHook</u>	Installs an application-defined hook function
<u>SetWindowsHookEx</u>	Installs an application-defined hook function
<u>UnhookWindowsHook</u>	Removes an application-defined hook function

### Icon Functions (3.1)

ArrangeIconicWindows

Arranges minimized child windows

CopyIcon

Copies an icon

CreateIcon

Creates an icon with the specified dimensions

DestroyIcon

Destroys an icon created by CreateIcon or LoadIcon

DrawIcon

Draws an icon in the specified device context

IsIconic

Determines whether a window is minimized

LoadIcon

Loads an icon resource

OpenIcon

Activates and displays a minimized window

### Information Functions (3.1)

<u>AnyPopup</u>	Indicates whether pop-up or overlapped window exists
<u>ChildWindowFromPoint</u>	Determines which child window contains a point
<u>EnumChildWindows</u>	Enumerates child windows
<u>EnumTaskWindows</u>	Enumerates windows associated with task on screen
<u>EnumWindows</u>	Enumerates parent windows
<u>FindWindow</u>	Returns window handle for class name and window name
<u>GetNextWindow</u>	Returns handle of window in window manager's list
<u>GetParent</u>	Returns parent window handle
<u>GetTopWindow</u>	Returns handle for top-level child of given window
<u>GetWindow</u>	Returns handle of window with specified relationship
<u>GetWindowTask</u>	Returns the task handle associated with a window
<u>IsChild</u>	Determines whether a window is a child
<u>IsValidWindow</u>	Determines whether a window handle is valid
<u>SetParent</u>	Changes a child's parent window
<u>WindowFromPoint</u>	Returns the handle of window containing a point



### Input Functions (3.1)

<u>EnableWindow</u>	Enables or disables input to a window or control
<u>GetActiveWindow</u>	Retrieves the handle of the active window
<u>GetCapture</u>	Returns the handle for the mouse-capture window
<u>GetCurrentTime</u>	Retrieves the elapsed time since Windows started
<u>GetDoubleClickTime</u>	Retrieves mouse double-click time
<u>GetFocus</u>	Returns handle of window with input focus
<u>GetTickCount</u>	Retrieves the amount of time Windows has been running
<u>IsWindowEnabled</u>	Determines whether a window accepts user input
<u>KillTimer</u>	Removes a timer
<u>ReleaseCapture</u>	Releases the mouse capture
<u>SetActiveWindow</u>	Makes a top-level window active
<u>SetCapture</u>	Sets the mouse capture to a window
<u>SetDoubleClickTime</u>	Sets the mouse double-click time
<u>SetFocus</u>	Sets the input focus to a window
<u>SetSysModalWindow</u>	Makes a window the system-modal window
<u>SetTimer</u>	Installs a system timer
<u>SwapMouseButton</u>	Reverses the meaning of mouse buttons

### Installable-Driver Functions (3.1)

<u>CloseDriver</u>	Closes an installable driver
<u>DefDriverProc</u>	Default processing of installable-driver messages
<u>DriverProc</u>	Processes installable-driver messages
<u>GetDriverModuleHandle</u>	Returns an installable-driver instance handle
<u>GetDriverInfo</u>	Retrieves installable-driver data
<u>GetNextDriver</u>	Enumerates installable-driver instances
<u>OpenDriver</u>	Opens an installable driver
<u>SendDriverMessage</u>	Sends a message to an installable driver

## Menu Functions (3.1)

A menu is an input tool in a Windows application that offers users one or more items, which they can select with the mouse or keyboard. An item in a menu bar can display a pop-up menu, and any item in a pop-up menu can display another pop-up menu. In addition, a pop-up menu can appear anywhere on the screen.

<u>AppendMenu</u>	Appends a new item to the end of a menu
<u>CheckMenuItem</u>	Changes a check mark by a menu item
<u>CreateMenu</u>	Creates a menu
<u>CreatePopupMenu</u>	Creates an empty pop-up window
<u>DeleteMenu</u>	Deletes an item from a menu
<u>DestroyMenu</u>	Destroys a menu
<u>DrawMenuBar</u>	Redraws the menu bar of a window
<u>EnableMenuItem</u>	Enables, disables, or grays a menu item
<u>GetMenu</u>	Returns a menu handle for the specified window
<u>GetMenuCheckMarkDimensions</u>	Retrieves default check mark bitmap dimensions
<u>GetMenuItemCount</u>	Retrieves the number of items in a menu
<u>GetMenuItemID</u>	Returns the handle of a menu item
<u>GetMenuState</u>	Retrieves status flags for a menu item
<u>GetMenuString</u>	Copies a menu-item label into a buffer
<u>GetSubMenu</u>	Returns a pop-up menu handle
<u>GetSystemMenu</u>	Provides access to the System menu
<u>HiliteMenuItem</u>	Changes highlighting of top-level menu item
<u>InsertMenu</u>	Inserts a new item in a menu
<u>LoadMenuIndirect</u>	Returns a menu handle for a menu template
<u>ModifyMenu</u>	Changes an existing menu item
<u>RemoveMenu</u>	Deletes a menu item and pop-up menu
<u>SetMenu</u>	Sets the menu for a window
<u>SetMenuItemBitmaps</u>	Associates bitmaps with a menu item
<u>TrackPopupMenu</u>	Displays and tracks a pop-up menu

### Message functions (3.1)

<u>CallWindowProc</u>	Passes message information to a window procedure
<u>DispatchMessage</u>	Dispatches a message to a window
<u>GetMessage</u>	Retrieves a message from the message queue
<u>GetMessagePos</u>	Retrieves the cursor position for the last message
<u>GetMessageTime</u>	Retrieves the time for the last message
<u>InSendMessage</u>	Determines whether window is processing SendMessage
<u>PeekMessage</u>	Checks an application's message queue
<u>PostAppMessage</u>	Posts a message to an application (task)
<u>PostMessage</u>	Places a message in a window's message queue
<u>PostQuitMessage</u>	Informs Windows that an application is exiting
<u>ReplyMessage</u>	Replies to a message sent through SendMessage
<u>SendMessage</u>	Sends a message to a window
<u>SetMessageQueue</u>	Creates a new message queue
<u>TranslateAccelerator</u>	Processes accelerator keys for menu commands
<u>TranslateMDISysAccel</u>	Processes MDI keyboard accelerators
<u>TranslateMessage</u>	Translates virtual-key messages
<u>WaitMessage</u>	Suspends an application and yields control

### Network functions (3.1)

<u>WNetAddConnection</u>	Adds a network connection
<u>WNetCancelConnection</u>	Removes a network connection
<u>WNetGetConnection</u>	Lists the network connection

### Painting Functions (3.1)

<u>BeginPaint</u>	Prepares a window for painting
<u>DrawFocusRect</u>	Draws a rectangle in the focus style
<u>DrawIcon</u>	Draws an icon in the specified device context
<u>EndPaint</u>	Marks the end of painting in the specified window
<u>ExcludeUpdateRgn</u>	Excludes an updated region from a clipping region
<u>FrameRect</u>	Draws a window border with the specified brush
<u>GetDC</u>	Returns a window device-context handle
<u>GetUpdateRect</u>	Retrieves window update-region dimensions
<u>GetUpdateRgn</u>	Retrieves the window update region
<u>GetWindowDC</u>	Retrieves the window device context
<u>GrayString</u>	Draws gray text at the specified location
<u>InvalidateRect</u>	Adds a rectangle to a window's update region
<u>InvalidateRgn</u>	Adds a region to a window's update region
<u>InvertRect</u>	Inverts a rectangular area
<u>ReleaseDC</u>	Frees a device context
<u>UpdateWindow</u>	Updates a window's client area
<u>ValidateRect</u>	Removes a rectangle from a window's update region
<u>ValidateRgn</u>	Removes a region from a window's update region

### Property Functions (3.1)

<u>EnumProps</u>	Enumerates property-list entries
<u>GetProp</u>	Returns a data handle from a window property list
<u>RemoveProp</u>	Removes a property-list entry
<u>SetProp</u>	Adds or changes a property-list entry

### Scrolling Functions (3.1)

<u>GetScrollPos</u>	Retrieves the current scroll-bar thumb position
<u>GetScrollRange</u>	Retrieves the minimum and maximum scroll-bar positions
<u>ScrollDC</u>	Scrolls a rectangle of bits horizontally and vertically
<u>ScrollWindow</u>	Scrolls the contents of a window's client area
<u>ScrollWindowEx</u>	Scrolls the contents of a window's client area
<u>SetScrollPos</u>	Sets the scroll-bar thumb position
<u>SetScrollRange</u>	Sets minimum and maximum scroll-bar positions
<u>ShowScrollBar</u>	Shows or hides a scroll bar



### System Functions (3.1)

<u>GetSysColor</u>	Retrieves the display-element color
<u>GetSystemMetrics</u>	Retrieves the system metrics
<u>GetTickCount</u>	Retrieves the amount of time Windows has been running
<u>SetSysColors</u>	Sets one or more system colors

### Window-Creation Functions (3.1)

<u>AdjustWindowRect</u>	Computes the required size of a window rectangle
<u>AdjustWindowRectEx</u>	Computes the required size of a window rectangle
<u>CreateWindow</u>	Creates an overlapped, pop-up, or child window
<u>CreateWindowEx</u>	Creates an overlapped, pop-up, or child window
<u>DefDlgProc</u>	Provides default window message processing
<u>DefFrameProc</u>	Provides default MDI frame window message processing
<u>DefMDIChildProc</u>	Provides default MDI child window message processing
<u>DefWindowProc</u>	Calls the default window procedure
<u>DestroyWindow</u>	Destroys a window
<u>GetClassInfo</u>	Returns window class information
<u>GetClassLong</u>	Retrieves a long value from extra class memory window class
<u>GetClassName</u>	Retrieves class name of a window
<u>GetClassWord</u>	Retrieves a word value from extra class memory window class memory word
<u>GetLastActivePopup</u>	Determines most recently active pop-up window
<u>GetWindowLong</u>	Retrieves a long value from extra window memory
<u>GetWindowWord</u>	Retrieves a word value from extra window memory
<u>RegisterClass</u>	Registers a window class
<u>SetClassLong</u>	Sets a long value in extra class memory
<u>SetClassWord</u>	Sets a word value in extra class memory
<u>SetWindowLong</u>	Sets a long value in extra window memory
<u>SetWindowWord</u>	Sets a word value in extra window memory
<u>UnregisterClass</u>	Frees a window class

### New 3.1 user functions

<u>CallNextHookEx</u>	Passes hook information down the hook chain
<u>CloseDriver</u>	Closes an installable driver
<u>CopyCursor</u>	Copies a cursor
<u>CopyIcon</u>	Copies an icon
<u>DefDriverProc</u>	Provides default processing of installable-driver messages
<u>EnableCommNotification</u>	Enables or disables <u>WM_COMMNOTIFY</u> posting
<u>EnableScrollBar</u>	Enables or disables scroll-bar arrows
<u>GetClipCursor</u>	Retrieves screen coordinates for cursor-confining rectangle
<u>GetCursor</u>	Returns current cursor handle
<u>GetDCEx</u>	Returns the handle of a device context
<u>GetDriverInfo</u>	Retrieves installable-driver data
<u>GetDriverModuleHandle</u>	Returns an installable-driver instance handle
<u>GetFreeSystemResources</u>	Returns percentage of free system resource space
<u>GetMessageExtraInfo</u>	Retrieves information about a hardware message
<u>GetNextDriver</u>	Enumerates installable-driver instances
<u>GetOpenClipboardWindow</u>	Returns handle to window that opened clipboard
<u>GetQueueStatus</u>	Determines queued message type
<u>GetSystemDebugState</u>	Returns system-state information to a debugger
<u>GetTimerResolution</u>	Retrieves the timer resolution
<u>GetWindowPlacement</u>	Retrieves window show state and min/max positions
<u>IsMenu</u>	Determines whether a menu handle is valid
<u>LockInput</u>	Locks input to all tasks except the current one
<u>LockWindowUpdate</u>	Disables or reenables drawing in a window
<u>MapWindowPoints</u>	Converts points to another coordinate system
<u>OpenDriver</u>	Opens an installable driver
<u>QuerySendMessage</u>	Determines whether message originated in task
<u>RedrawWindow</u>	Updates rectangle or region in window's client area
<u>ScrollWindowEx</u>	Scrolls the contents of a window's client area
<u>SendDriverMessage</u>	Sends a message to an installable driver
<u>SetWindowPlacement</u>	Sets window show state and min/max position
<u>SetWindowsHookEx</u>	Installs an application defined hook function
<u>SubtractRect</u>	Creates a rectangle from the difference of two
<u>SystemParametersInfo</u>	Queries or sets system-wide parameters
<u>UnhookWindowsHookEx</u>	Removes a function from the hook chain
<u>WNetAddConnection</u>	Adds a network connection
<u>WNetCancelConnection</u>	Removes a network connection
<u>WNetGetConnection</u>	Lists a network connection

### Bitmap Functions (3.1)

<u>BitBlt</u>	Copies a bitmap between device contexts
<u>CreateBitmap</u>	Creates a device-dependent memory bitmap
<u>CreateBitmapIndirect</u>	Creates a bitmap using a <u>BITMAP</u> structure
<u>CreateCompatibleBitmap</u>	Creates a bitmap compatible with a device context
<u>CreateDIBitmap</u>	Creates a bitmap handle from DIB specification
<u>CreateDiscardableBitmap</u>	Creates a discardable bitmap
<u>GetBitmapBits</u>	Copies bitmap bits into a buffer
<u>GetBitmapDimension</u>	Retrieves the width and height of a bitmap
<u>GetBitmapDimensionEx</u>	Retrieves the width and height of a bitmap
<u>GetDIBits</u>	Copies DIB bits into a buffer
<u>GetPixel</u>	Retrieves the <u>RGB</u> color of a specified pixel
<u>LoadBitmap</u>	Loads a bitmap resource
<u>PatBlt</u>	Creates a bit pattern on a device
<u>SetBitmapBits</u>	Sets bitmap bits from an array of bytes
<u>SetBitmapDimension</u>	Sets the width and height of bitmap
<u>SetBitmapDimensionEx</u>	Sets the width and height of bitmap
<u>SetDIBits</u>	Sets the bits of a bitmap
<u>SetDIBitsToDevice</u>	Sets DIB bits to a device
<u>SetPixel</u>	Sets a pixel to a specified color
<u>StretchBlt</u>	Sets the bitmap-stretching mode
<u>StretchDIBits</u>	Moves DIB from source to destination rectangle

### Clipping Functions (3.1)

<u>ExcludeClipRect</u>	Creates new clipping region, excluding rectangle
<u>GetClipBox</u>	Retrieves a rectangle for the clipping region
<u>IntersectClipRect</u>	Creates a clipping region from an intersection
<u>OffsetClipRgn</u>	Moves a clipping region
<u>PtVisible</u>	Queries whether a point is within the clipping region
<u>RectVisible</u>	Queries whether a rectangle is within the clipping region
<u>SelectClipRgn</u>	Selects a clipping region for the device context

### Coordinate Functions (3.1)

<u>ChildWindowFromPoint</u>	Determines which child window contains a point
<u>ClientToScreen</u>	Converts client point to screen coordinates
<u>DPtoLP</u>	Converts device points to logical coordinates
<u>GetCurrentPosition</u>	Retrieves position in logical coordinates
<u>GetCurrentPositionEx</u>	Retrieves position in logical coordinates
<u>LPtoDP</u>	Converts logical points to device coordinates
<u>MapWindowPoints</u>	Converts points to another coordinate system
<u>ScreenToClient</u>	Converts screen points to client coordinates
<u>WindowFromPoint</u>	Returns the handle of a window containing a point

### Device-Context Functions (3.1)

<u>CreateCompatibleDC</u>	Creates a DC compatible with a specified device
<u>CreateDC</u>	Creates a device context
<u>CreateIC</u>	Creates an information context
<u>DeleteDC</u>	Deletes a device context
<u>GetDC</u>	Retrieves the handle of a device context
<u>GetDCEx</u>	Retrieves the handle of a device context
<u>GetDCOrg</u>	Retrieves the translation origin for a device context
<u>ResetDC</u>	Updates a device context
<u>RestoreDC</u>	Restores a device context
<u>ReleaseDC</u>	Frees a device context
<u>SaveDC</u>	Saves the current state of a device context

### Drawing-Attribute Functions (3.1)

<u>GetBkColor</u>	Retrieves the current background color
<u>GetBkMode</u>	Retrieves the background mode
<u>GetPolyFillMode</u>	Retrieves the current polygon-filling mode
<u>GetROP2</u>	Retrieves the current drawing mode
<u>GetStretchBltMode</u>	Retrieves the current bitmap-stretching mode
<u>GetTextColor</u>	Retrieves the current text color
<u>SetBkColor</u>	Sets the background color
<u>SetBkMode</u>	Sets the background mode
<u>SetPolyFillMode</u>	Sets the polygon-filling mode
<u>SetROP2</u>	Sets the drawing mode
<u>SetStretchBltMode</u>	Sets the bitmap-stretching mode
<u>SetTextColor</u>	Sets the foreground color of text



### Drawing-Tool Functions (3.1)

<u>CreateBrushIndirect</u>	Creates a brush with the specified attributes
<u>CreateDIBPatternBrush</u>	Creates a pattern brush from a DIB
<u>CreateHatchBrush</u>	Creates a hatched brush
<u>CreatePatternBrush</u>	Creates a pattern brush from a bitmap
<u>CreatePen</u>	Creates a pen with the specified attributes
<u>CreatePenIndirect</u>	Creates a pen using a LOGPEN structure
<u>CreateSolidBrush</u>	Creates a solid brush with a specified color
<u>DeleteObject</u>	Deletes an object from memory
<u>EnumObjects</u>	Enumerates the pens and brushes in a device context
<u>GetBrushOrg</u>	Retrieves the origin of the current brush
<u>GetObject</u>	Retrieves information about an object
<u>GetStockObject</u>	Retrieves the handle of a stock pen, brush, or font
<u>IsGDIObject</u>	Determines if handle is not handle of GDI object
<u>SelectObject</u>	Selects an object into a device context
<u>SetBrushOrg</u>	Sets the origin of the current brush
<u>UnrealizeObject</u>	Resets brush origins or logical palettes

## Ellipse and Polygon Functions (3.1)

<u>Chord</u>	Draws a chord
<u>DrawFocusRect</u>	Draws a rectangle in the focus style
<u>Ellipse</u>	Draws an ellipse
<u>Pie</u>	Draws a pie-shaped wedge
<u>Polygon</u>	Draws a polygon
<u>PolyPolygon</u>	Draws a series of polygons
<u>Rectangle</u>	Draws a rectangle
<u>RoundRect</u>	Draws a rectangle with rounded corners

### Font Functions (3.1)

<u>AddFontResource</u>	Adds a font resource to the font table
<u>CreateFont</u>	Creates a logical font
<u>CreateFontIndirect</u>	Creates a font using the LOGFONT structure
<u>CreateScalableFontResource</u>	Creates a resource file with font information
<u>EnumFontFamilies</u>	Enumerates fonts in a specified family
<u>EnumFonts</u>	Enumerates fonts on a specified device
<u>GetAspectRatioFilter</u>	Retrieves the current aspect-ratio filter
<u>GetAspectRatioFilterEx</u>	Retrieves the current aspect-ratio filter
<u>GetCharABCWidths</u>	Retrieves the widths of consecutive characters
<u>GetCharWidth</u>	Retrieves character widths
<u>GetFontData</u>	Retrieves font-metric information
<u>GetGlyphOutline</u>	Retrieves data for individual outline character
<u>GetKerningPairs</u>	Retrieves kerning pairs for the current font
<u>GetOutlineTextMetrics</u>	Retrieves metrics for TrueType fonts
<u>GetRasterizerCaps</u>	Retrieves status of TrueType fonts on system
<u>RemoveFontResource</u>	Removes an added font resource
<u>SetMapperFlags</u>	Sets the font-mapper flag

### Line-Output Functions (3.1)

<u>Arc</u>	Draws an arc
<u>LineDDA</u>	Computes successive points in a line
<u>LineTo</u>	Draws a line from the current position
<u>MoveTo</u>	Moves the current position
<u>MoveToEx</u>	Moves the current position
<u>Polyline</u>	Draws line segments to connect specified points

### Mapping Functions (3.1)

<u>GetMapMode</u>	Retrieves the mapping mode
<u>GetViewportExt</u>	Retrieves viewport extents
<u>GetViewportExtEx</u>	Retrieves viewport extents
<u>GetViewportOrg</u>	Retrieves the viewport origin
<u>GetViewportOrgEx</u>	Retrieves the viewport origin
<u>GetWindowExt</u>	Retrieves the window extents
<u>GetWindowExtEx</u>	Retrieves the window extents
<u>GetWindowOrg</u>	Retrieves the window origin
<u>GetWindowOrgEx</u>	Retrieves the window origin
<u>OffsetViewportOrg</u>	Moves the viewport origin
<u>OffsetViewportOrgEx</u>	Moves the viewport origin
<u>OffsetWindowOrg</u>	Moves the window origin
<u>OffsetWindowOrgEx</u>	Moves the window origin
<u>ScaleViewportExt</u>	Scales the viewport extents
<u>ScaleViewportExtEx</u>	Scales the viewport extents
<u>ScaleWindowExt</u>	Scales the window extents
<u>ScaleWindowExtEx</u>	Scales the window extents
<u>SetMapMode</u>	Sets the mapping mode
<u>SetViewportExt</u>	Sets the viewport extents
<u>SetViewportExtEx</u>	Sets the viewport extents
<u>SetViewportOrg</u>	Sets the viewport origin
<u>SetViewportOrgEx</u>	Sets the viewport origin
<u>SetWindowExt</u>	Sets the window extents
<u>SetWindowExtEx</u>	Sets the window extents
<u>SetWindowOrg</u>	Sets the window origin
<u>SetWindowOrgEx</u>	Sets the window origin

### Metafile Functions (3.1)

<u>CloseMetaFile</u>	Closes a metafile device context and creates a handle
<u>CopyMetaFile</u>	Copies a source metafile to a file
<u>CreateMetaFile</u>	Creates a metafile device context
<u>DeleteMetaFile</u>	Invalidates a metafile handle
<u>EnumMetaFile</u>	Enumerates the metafile records in a metafile
<u>GetMetaFile</u>	Creates a handle to a metafile
<u>GetMetaFileBits</u>	Creates a memory object from a metafile
<u>PlayMetaFile</u>	Plays a metafile
<u>PlayMetaFileRecord</u>	Plays a metafile record
<u>SetMetaFileBits</u>	Creates a memory object from a metafile
<u>SetMetaFileBitsBetter</u>	Creates a memory object from a metafile

### Palette Functions (3.1)

<u>AnimatePalette</u>	Replaces entries in a logical palette
<u>CreatePalette</u>	Creates a logical color palette
<u>GetNearestColor</u>	Retrieves the closest available color
<u>GetNearestPaletteIndex</u>	Retrieves the nearest match for a color
<u>GetPaletteEntries</u>	Retrieves a range of palette entries
<u>GetSystemPaletteEntries</u>	Retrieves entries from the system palette
<u>GetSystemPaletteUse</u>	Determines access to the entire system palette
<u>RealizePalette</u>	Maps entries from logical to system palette
<u>ResizePalette</u>	Changes the size of a logical palette
<u>SelectPalette</u>	Selects a palette into a device context
<u>SetPaletteEntries</u>	Sets colors and flags for a logical palette
<u>SetSystemPaletteUse</u>	Sets the use of static colors in the system palette

### Printer-Control Functions (3.1)

<u>AbortDoc</u>	Terminates a print job
<u>DeviceCapabilities</u>	Retrieves the capabilities of a device
<u>DeviceMode</u>	Displays a dialog box for the printing modes
<u>EndDoc</u>	Ends a print job
<u>EndPage</u>	Ends a page
<u>Escape</u>	Allows access to device capabilities
<u>ExtDeviceMode</u>	Displays a dialog box for the printing modes
<u>GetDeviceCaps</u>	Retrieves the device capabilities
<u>SetAbortProc</u>	Sets the abort function for a print job
<u>SpoolFile</u>	Puts a file in the spooler queue
<u>StartDoc</u>	Starts a print job
<u>StartPage</u>	Prepares the printer driver to accept data
<u>QueryAbort</u>	Queries whether to terminate a print job



### Rectangle Functions (3.1)

<u>CopyRect</u>	Copies the dimensions of a rectangle
<u>EqualRect</u>	Determines whether two rectangles are equal
<u>FrameRect</u>	Draws a window border with the specified brush
<u>FillRect</u>	Fills a rectangle with the specified brush
<u>GetBoundsRect</u>	Returns current accumulated bounding rectangle
<u>InflateRect</u>	Changes rectangle dimensions
<u>IntersectRect</u>	Calculates the intersection of two rectangles
<u>InvertRect</u>	Inverts a rectangular area
<u>IsEmptyRect</u>	Determines whether a rectangle is empty
<u>OffsetRect</u>	Moves a rectangle by the specified offsets
<u>PtInRect</u>	Determines whether a point is in a rectangle
<u>SetBoundsRect</u>	Controls bounding-rectangle accumulation
<u>SetRect</u>	Sets rectangle coordinates
<u>SetRectEmpty</u>	Creates an empty rectangle
<u>SubtractRect</u>	Creates rectangle from difference of two others
<u>UnionRect</u>	Creates the union of two rectangles

### Region Functions (3.1)

<u>CombineRgn</u>	Creates a region by combining two regions
<u>CreateEllipticRgn</u>	Creates an elliptical region
<u>CreateEllipticRgnIndirect</u>	Creates an elliptical region
<u>CreatePolygonRgn</u>	Creates a polygonal region
<u>CreatePolyPolygonRgn</u>	Creates a region consisting of polygons
<u>CreateRectRgn</u>	Creates a rectangular region
<u>CreateRectRgnIndirect</u>	Creates a region using a <u>RECT</u> structure
<u>CreateRoundRectRgn</u>	Creates a rectangular region with round corners
<u>EqualRgn</u>	Compares two regions for equality
<u>FillRgn</u>	Fills a region with the specified brush
<u>FrameRgn</u>	Draws a border around a region
<u>GetRgnBox</u>	Retrieves the bounding rectangle for a region
<u>InvertRgn</u>	Inverts the colors in a region
<u>OffsetRgn</u>	Moves a region by the specified offsets
<u>PaintRgn</u>	Fills region with brush in device context
<u>PtInRegion</u>	Queries whether a point is in a region
<u>RectInRegion</u>	Queries whether a rectangle overlaps a region
<u>SetRectRgn</u>	Changes a region into the specified rectangle

### Text Functions (3.1)

<u>DrawText</u>	Writes a character string in a rectangular region
<u>ExtTextOut</u>	Determines the dimensions of a tabbed string
<u>GetTabbedTextExtent</u>	Retrieves the status of text-alignment flags
<u>GetTextAlign</u>	Retrieves the intercharacter spacing
<u>GetTextCharacterExtra</u>	Computes the dimensions of a string
<u>GetTextExtent</u>	Computes the dimensions of a string
<u>GetTextExtentPoint</u>	Retrieves the typeface name of the current font
<u>GetTextFace</u>	Retrieves the metrics for the current font
<u>GetTextMetrics</u>	Sets text-alignment flags for the device context
<u>SetTextAlign</u>	Sets the intercharacter spacing
<u>SetTextCharacterExtra</u>	Sets the justification for text output
<u>SetTextJustification</u>	Writes a tabbed character string
<u>TabbedTextOut</u>	Writes a character string at the specified location
<u>TextOut</u>	

### TrueType Functions (3.1)

CreateScalableFontResource

Creates a resource file with font information

GetCharABCWidths

Retrieves the widths of consecutive characters

GetFontData

Retrieves font-metric information

GetGlyphOutline

Retrieves data for individual outline character

GetKerningPairs

Retrieves kerning pairs for the current font

GetOutlineTextMetrics

Retrieves metric information for TrueType fonts

GetRasterizerCaps

Retrieves status of TrueType fonts on system

### Pen Functions (3.1)

<u>CreatePen</u>	Creates a pen with the specified attributes
<u>CreatePenIndirect</u>	Creates a pen using a LOGPEN structure
<u>GetStockObject</u>	Retrieves the handle of a stock pen, brush, or font
<u>LineTo</u>	Draws a line from the current position

### Brush Functions (3.1)

<u>CreateBrushIndirect</u>	Creates a brush with the specified attributes
<u>CreateHatchBrush</u>	Creates a hatched brush
<u>CreatePatternBrush</u>	Creates a pattern brush from a bitmap
<u>CreateSolidBrush</u>	Creates a solid brush with a specified color
<u>GetBrushOrg</u>	Retrieves the origin of the current brush
<u>GetBrushOrgEx</u>	Retrieves the origin of the current brush
<u>GetStockObject</u>	Retrieves the handle of a stock pen, brush, or font
<u>SetBrushOrg</u>	Sets the origin of the current brush

### New 3.1 GDI functions

<u>AbortDoc</u>	Terminates a print job
<u>CreateScalableFontResource</u>	Creates a resource file with font information
<u>EndDoc</u>	Ends a print job
<u>EndPage</u>	Ends a page
<u>EnumFontFamilies</u>	Enumerates fonts in a specified family
<u>GetAspectRatioFilterEx</u>	Retrieves the current aspect-ratio filter
<u>GetBoundsRect</u>	Returns current accumulated bounding rectangle
<u>GetBrushOrgEx</u>	Retrieves the origin of the current brush
<u>GetCharABCWidths</u>	Retrieves the widths of consecutive characters
<u>GetCurrentPositionEx</u>	Retrieves the current position in logical units
<u>GetFontData</u>	Retrieves font-metric information
<u>GetGlyphOutline</u>	Retrieves data for individual outline character
<u>GetKerningPairs</u>	Retrieves kerning pairs for the current font
<u>GetOutlineTextMetrics</u>	Retrieves metrics for TrueType fonts
<u>GetRasterizerCaps</u>	Retrieves status of TrueType fonts on system
<u>GetTextExtentPoint</u>	Computes the dimensions of a string
<u>GetViewportExtEx</u>	Retrieves the viewport extents
<u>GetViewportOrgEx</u>	Retrieves the viewport origin
<u>GetWindowExtEx</u>	Retrieves the window extents
<u>GetWindowOrgEx</u>	Retrieves the window origin
<u>IsGDIObject</u>	Determines whether handle is not a <u>GDI</u> object
<u>MoveToEx</u>	Moves the current position
<u>OffsetViewportOrgEx</u>	Moves the viewport origin
<u>OffsetWindowOrgEx</u>	Moves the window origin
<u>QueryAbort</u>	Queries whether to terminate a print job
<u>ResetDC</u>	Updates a device context
<u>ScaleViewportExtEx</u>	Scales the viewport extents
<u>ScaleWindowExtEx</u>	Scales the window extents
<u>SetAbortProc</u>	Sets the abort function for a print job
<u>SetBitmapDimensionEx</u>	Sets the width and height of a bitmap
<u>SetBoundsRect</u>	Controls bounding-rectangle accumulation
<u>SetMetaFileBitsBetter</u>	Creates a memory object from a metafile
<u>SetViewportExtEx</u>	Sets the viewport extents
<u>SetViewportOrgEx</u>	Sets the viewport origin
<u>SetWindowExtEx</u>	Sets the window extents
<u>SetWindowOrgEx</u>	Sets the window origin
<u>SpoolFile</u>	Puts a file in the spooler queue
<u>StartDoc</u>	Starts a print job
<u>StartPage</u>	Prepares a printer driver to accept data

### Application Execution Functions (3.1)

<u>LoadModule</u>	Loads and executes Windows application
<u>WinExec</u>	Runs the specified application
<u>WinHelp</u>	Invokes Windows Help



## Atom-management Functions (3.1)

Atom-management functions create and manipulate atoms. Atoms are integers that uniquely identify character strings. They are useful in applications that use many character strings and in applications that need to conserve memory. Windows stores atoms in atom tables. A local atom table is allocated in an application's data segment; it cannot be accessed by other applications. The global atom table can be shared and is useful in applications that use dynamic data exchange (DDE).

<u>AddAtom</u>	Adds a string to the local atom table
<u>DeleteAtom</u>	Decrements the reference count of a local atom
<u>FindAtom</u>	Retrieves a string atom from a local atom table
<u>GetAtomHandle</u>	Retrieves an atom handle
<u>GetAtomName</u>	Retrieves a local atom string
<u>GlobalAddAtom</u>	Adds a string to the system atom table
<u>GlobalDeleteAtom</u>	Decrements the reference count of a global atom
<u>GlobalFindAtom</u>	Retrieves a string atom from a global atom table
<u>GlobalGetAtomName</u>	Retrieves a global atom string
<u>InitAtomTable</u>	Sets the size of the local atom hash table

### Communication Functions (3.1)

<u>BuildCommDCB</u>	Translates a device-definition string to a <u>DCB</u>
<u>ClearCommBreak</u>	Restores character transmission
<u>CloseComm</u>	Closes a communications device
<u>EnableCommNotification</u>	Enables or disables <u>WM_COMMNOTIFY</u> posting
<u>EscapeCommFunction</u>	Passes an extended <u>function</u> to a device
<u>FlushComm</u>	Flushes a transmission or receiving queue
<u>GetCommError</u>	Retrieves the communications-device status
<u>GetCommEventMask</u>	Retrieves the device event word
<u>GetCommState</u>	Retrieves the device control block
<u>OpenComm</u>	Opens a communications device
<u>ReadComm</u>	Reads from a communications device
<u>SetCommBreak</u>	Suspends character transmission
<u>SetCommEventMask</u>	Enables events in a device event word
<u>SetCommState</u>	Sets the communications-device state
<u>TransmitCommChar</u>	Places a character in the transmission queue
<u>UngetCommChar</u>	Puts a character back in the receiving queue
<u>WriteComm</u>	Writes to a communications device

### Debugging Functions (3.1)

<u>DebugBreak</u>	Causes a breakpoint exception
<u>DebugOutput</u>	Sends messages to the debugging terminal
<u>DirectedYield</u>	Forces execution of a specified task
<u>FatalAppExit</u>	Terminates an application
<u>FatalExit</u>	Sends current state of Windows to the debugger
<u>GetSystemDebugState</u>	Returns system-state information to a debugger
<u>GetWinDebugInfo</u>	Retrieves current system-debugging information
<u>LockInput</u>	Locks input to all tasks except the current one
<u>LogError</u>	Identifies the most recent system error
<u>LogParamError</u>	Identifies a parameter validation error
<u>OutputDebugString</u>	Sends a character string to the debugger
<u>QuerySendMessage</u>	Determines whether a message originated in a task
<u>SetWinDebugInfo</u>	Sets the current system-debugging information
<u>ValidateCodeSegments</u>	Tests for random memory overwrites
<u>ValidateFreeSpaces</u>	Checks free memory for valid contents

### File I/O Functions (3.1)

<u>GetDriveType</u>	Determines the drive type
<u>GetSystemDirectory</u>	Returns the path of the Windows system directory
<u>GetTempDrive</u>	Returns a disk drive letter for temporary files
<u>GetTempFileName</u>	Creates a temporary filename
<u>GetWindowsDirectory</u>	Returns the path of the Windows directory
<u>_hread</u>	Reads data from a file
<u>_hwrite</u>	Writes data to a file
<u>_lclose</u>	Closes an open file
<u>_lcreat</u>	Creates or opens a file
<u>_llseek</u>	Repositions the file pointer
<u>_lopen</u>	Opens an existing file
<u>_lread</u>	Reads data from a file
<u>_lwrite</u>	Writes data to a file
<u>hmemcpy</u>	Copies bytes from source to destination buffer
<u>OpenFile</u>	Creates, opens, reopens, or deletes a file
<u>SetHandleCount</u>	Changes the number of available file handles

### Initialization-File Functions (3.1)

Initialization-file functions obtain information from and copy information to a Windows or private (application-specific) initialization file. The Windows initialization file (WIN.INI) is a special ASCII file that contains entry-value pairs representing run-time options for applications.

An application should use a private initialization file to record information that affects it alone. This improves the performance of the application and Windows by reducing the amount of information that Windows must read when it accesses the initialization file. An application should record information in WIN.INI only if the information affects the Windows environment or other applications and should send the WM\_WININICHANGE message to all top-level windows.

The WININI.WRI and SYSINI.WRI files supplied with the retail version of Windows describe the contents of the WIN.INI and SYSTEM.INI files, respectively.

<u>GetPrivateProfileInt</u>	Retrieves integer value from initialization file
<u>GetPrivateProfileString</u>	Retrieves a string from an initialization file
<u>GetProfileInt</u>	Retrieves an integer value from WIN.INI
<u>GetProfileString</u>	Retrieves a string from WIN.INI
<u>WritePrivateProfileString</u>	Writes a string to an initialization file
<u>WriteProfileString</u>	Writes a string to WIN.INI

## Memory-Management Functions (3.1)

Memory-management functions manage system memory. There are two categories of memory-management functions: those that manage global memory and those that manage local memory. Global memory is all memory in the system that has not been allocated by an application or reserved by the system. Local memory is the memory in the data segment of a Windows application.

<u>GetFreeSpace</u>	Returns the number of free bytes in the global heap
<u>GetFreeSystemResources</u>	Returns the percentage of free system-resource space
<u>GetSelectorBase</u>	Retrieves the base address of a selector
<u>GetSelectorLimit</u>	Retrieves the limit of a selector
<u>GetWinFlags</u>	Returns the current system configuration flags
<u>GlobalAlloc</u>	Allocates memory from the global heap
<u>GlobalCompact</u>	Generates free global memory by compacting
<u>GlobalDosAlloc</u>	Allocates memory available to MS-DOS in real mode
<u>GlobalDosFree</u>	Frees global memory allocated by GlobalDosAlloc
<u>GlobalFlags</u>	Returns information about a global memory object
<u>GlobalFree</u>	Frees a global memory object
<u>GlobalHandle</u>	Retrieves the handle for a specified selector
<u>GlobalLock</u>	Locks global memory object and returns pointer
<u>GlobalLRUNewest</u>	Moves global memory object to newest LRU position
<u>GlobalLRUOldest</u>	Moves global memory object to oldest LRU position
<u>GlobalNotify</u>	Installs a notification procedure
<u>GlobalReAlloc</u>	Changes size or attributes of global memory object
<u>GlobalSize</u>	Returns the size of a global memory object
<u>GlobalUnlock</u>	Unlocks a global memory object
<u>GlobalUnwire</u>	Not used in Windows 3.1
<u>GlobalWire</u>	Not used in Windows 3.1
<u>LimitEMSPages</u>	Not used in Windows 3.1
<u>LocalAlloc</u>	Allocates memory from the local heap
<u>LocalCompact</u>	Generates free local memory by compacting
<u>LocalFlags</u>	Returns local memory object information
<u>LocalFree</u>	Frees a local memory object
<u>LocalHandle</u>	Returns the handle of a local memory object
<u>LocalInit</u>	Initializes the specified local heap
<u>LocalLock</u>	Locks local memory object and returns pointer
<u>LocalReAlloc</u>	Changes size or attributes of local memory object
<u>LocalShrink</u>	Shrinks the specified local heap
<u>LocalSize</u>	Returns the size of a local memory object
<u>LocalUnlock</u>	Unlocks a local memory object
<u>LockSegment</u>	Locks a discardable memory segment
<u>SetSelectorBase</u>	Sets the base address of a selector
<u>SetSelectorLimit</u>	Sets the limit of a selector
<u>SetSwapAreaSize</u>	Sets the amount of memory used for code segments
<u>SwitchStackBack</u>	Restores the current task stack
<u>SwitchStackTo</u>	Changes the location of the current task stack
<u>UnLockSegment</u>	Unlocks a discardable memory segment

## Module-Management Functions (3.1)

Module-management functions alter and retrieve information about Windows modules, which are loadable, executable units of code and data.

<u>FreeLibrary</u>	Frees a loaded library module
<u>FreeModule</u>	Frees a loaded module
<u>FreeProcInstance</u>	Frees a function instance
<u>GetCodeHandle</u>	Determines the location of a function
<u>GetInstanceData</u>	Copies data from previous instance to current one
<u>GetModuleFileName</u>	Returns the filename for a module
<u>GetModuleHandle</u>	Retrieves a handle for the specified module
<u>GetModuleUsage</u>	Retrieves the reference count of a module
<u>GetProcAddress</u>	Returns the address of an exported DLL function
<u>GetVersion</u>	Returns the current MS-DOS and Windows versions
<u>LoadLibrary</u>	Loads the specified library module
<u>MakeProcInstance</u>	Returns address of prolog code for function

### Optimization-Tool Functions (3.1)

<u>ProfClear</u>	Discards all buffered Profiler samples
<u>ProfFinish</u>	Stops Profiler sampling and flushes the buffer
<u>ProfFlush</u>	Flushes the Profiler sampling buffer to a disk
<u>ProfInsChk</u>	Determines whether Profiler is installed
<u>ProfSampRate</u>	Sets the Profiler code-sampling rate
<u>ProfSetup</u>	Sets the Profiler buffer size and sample quantity
<u>ProfStart</u>	Starts Profiler sampling
<u>ProfStop</u>	Stops Profiler sampling
<u>SwapRecording</u>	Starts or stops recording of memory swapping



### Pointer-Validation Functions (3.1)

<u>IsBadCodePtr</u>	Determines whether a code pointer is valid
<u>IsBadHugeReadPtr</u>	Determines whether a huge read pointer is valid
<u>IsBadHugeWritePtr</u>	Determines whether a huge write pointer is valid
<u>IsBadReadPtr</u>	Determines whether a read pointer is valid
<u>IsBadStringPtr</u>	Determines whether a string pointer is valid
<u>IsBadWritePtr</u>	Determines whether a write pointer is valid

## Resource-Management Functions (3.1)

Resource-management functions find and load application resources from a Windows executable file. A resource can be a cursor, icon, bitmap, string, or font.

<u>AccessResource</u>	Opens an executable file and locates a resource
<u>AllocResource</u>	Allocates memory for a resource
<u>FindResource</u>	Locates a resource in a resource file
<u>FreeResource</u>	Frees a loaded resource
<u>LoadAccelerators</u>	Loads an accelerator table
<u>LoadBitmap</u>	Loads a bitmap resource
<u>LoadCursor</u>	Loads a cursor resource
<u>LoadIcon</u>	Loads an icon resource
<u>LoadMenu</u>	Loads a menu resource
<u>LoadResource</u>	Loads the specified resource in global memory
<u>LoadString</u>	Loads a string resource
<u>LockResource</u>	Locks a resource in memory
<u>SetResourceHandler</u>	Installs a callback function that loads resources
<u>SizeofResource</u>	Returns the size of a resource

## Segment Functions (3.1)

Segment functions allocate, free, and convert selectors; lock and unlock memory objects referenced by selectors; and retrieve information about segments.

An application should not use these functions unless it is absolutely necessary. Use of these functions violates preferred Windows programming practices.

<u>AllocDStoCSAlias</u>	Translates a data segment to a code segment
<u>AllocSelector</u>	Allocates a new selector
<u>FreeSelector</u>	Frees an allocated selector
<u>GetCodeInfo</u>	Retrieves code-segment information
<u>GlobalFix</u>	Locks a global memory object in linear memory
<u>GlobalPageLock</u>	Increments the global memory page-lock count
<u>GlobalPageUnlock</u>	Decrements the global memory page-lock count
<u>GlobalUnfix</u>	Unlocks a global memory object in linear memory
<u>LockSegment</u>	Locks a discardable memory segment
<u>PrestoChangoSelector</u>	Converts a code or data selector
<u>UnlockSegment</u>	Unlocks a discardable memory segment

## String-Manipulation Functions (3.1)

String-manipulation functions translate strings from one character set to another, determine and convert the case of strings, determine whether a character is alphabetic or alphanumeric, find adjacent characters in a string, and perform other string manipulations.

<u>AnsiLower</u>	Converts a string to lowercase
<u>AnsiLowerBuff</u>	Converts a buffer string to lowercase
<u>AnsiNext</u>	Moves to the next character in a string
<u>AnsiPrev</u>	Moves to the previous character in a string
<u>AnsiToOem</u>	Translates a Windows string to an OEM string
<u>AnsiToOemBuff</u>	Translates a Windows string to an OEM string
<u>AnsiUpper</u>	Converts a string to uppercase
<u>AnsiUpperBuff</u>	Converts a buffer string to uppercase
<u>IsCharAlpha</u>	Determines whether a character is alphabetic
<u>IsCharAlphaNumeric</u>	Determines whether a character is alphanumeric
<u>IsCharLower</u>	Determines whether a character is lowercase
<u>IsCharUpper</u>	Determines whether a character is uppercase
<u>lstrcat</u>	Appends one string to another
<u>lstrcmp</u>	Compares two character strings
<u>lstrcmpi</u>	Compares two character strings
<u>lstrcpy</u>	Copies a string to a buffer
<u>lstrcpyn</u>	Copies characters from a string to a buffer
<u>lstrlen</u>	Returns the length, in bytes, of a string
<u>OemToAnsi</u>	Translates an OEM string to a Windows string
<u>OemToAnsiBuff</u>	Translates an OEM string to a Windows string
<u>ToAscii</u>	Translates virtual-key code to Windows character
<u>wsprintf</u>	Formats and stores a string in a buffer
<u>wvsprintf</u>	Formats and stores a string in a buffer

## Task Functions (3.1)

Task functions alter the execution status of tasks, return information associated with a task, and retrieve information about the environment in which the task is being executed. A task is a single Windows application call.

<u>Catch</u>	Captures the current execution environment
<u>ExitWindows</u>	Restarts or terminates Windows
<u>GetCurrentPDB</u>	Returns the selector address of the current PDB
<u>GetCurrentTask</u>	Returns the current task handle
<u>GetDOSEnvironment</u>	Returns a far pointer to the current environment
<u>GetNumTasks</u>	Retrieves the current number of tasks
<u>IsTask</u>	Determines whether a task handle is valid
<u>SetErrorMode</u>	Controls Interrupt 24h error handling
<u>Throw</u>	Restores the execution environment
<u>Yield</u>	Stops the current task

### New 3.1 kernel functions

<u>_hread</u>	Reads data from a file
<u>_hwrite</u>	Writes data to a file
<u>DebugOutput</u>	Sends messages to the debugging terminal
<u>DirectedYield</u>	Forces execution to continue with specified task
<u>GetSelectorBase</u>	Retrieves the base address of a selector
<u>GetSelectorLimit</u>	Retrieves the limit of a selector
<u>GetWinDebugInfo</u>	Queries current system-debugging information
<u>hmemcpy</u>	Copies bytes from source to destination buffer
<u>IsBadCodePtr</u>	Determines whether a code pointer is valid
<u>IsBadHugeReadPtr</u>	Determines whether huge read pointer is valid
<u>IsBadHugeWritePtr</u>	Determines whether a huge write pointer is valid
<u>IsBadReadPtr</u>	Determines whether a read pointer is valid
<u>IsBadStringPtr</u>	Determines whether a string pointer is valid
<u>IsBadWritePtr</u>	Determines whether a write pointer is valid
<u>IsDBCSLeadByte</u>	Determines whether character is DBCS lead byte
<u>IsTask</u>	Determines whether a task handle is valid
<u>LogError</u>	Identifies a system error
<u>LogParamError</u>	Identifies a parameter validation error
<u>SetSelectorBase</u>	Sets the base address of a selector
<u>SetSelectorLimit</u>	Sets the limit of a selector
<u>SetWinDebugInfo</u>	Sets the current system-debugging information

OLE function groups (3.1)

Document Functions

Link Functions

Object-creation Functions

Object-management Functions

Server Functions (Client)

Server Functions (Server)

### Document Functions (3.1)

<u>OleEnumObjects</u>	Enumerates objects in a document
<u>OleRegisterClientDoc</u>	Registers a document with the library
<u>OleRegisterServerDoc</u>	Registers a document with the server library
<u>OleRename</u>	Informs the library that an object is renamed
<u>OleRenameClientDoc</u>	Informs the library that a document is renamed
<u>OleRenameServerDoc</u>	Informs the library that a document is renamed
<u>OleRevertClientDoc</u>	Informs library that document reverted to saved state
<u>OleRevertServerDoc</u>	Informs library that document is reset to saved state
<u>OleRevokeClientDoc</u>	Informs the library that a document is not open
<u>OleRevokeServerDoc</u>	Revokes the specified document
<u>OleSavedClientDoc</u>	Informs library that a document has been saved
<u>OleSavedServerDoc</u>	Informs library that a document has been saved



### Link Functions (3.1)

OleGetLinkUpdateOptions

Retrieves update options for an object

OleQueryLinkFromClip

Retrieves link data for clipboard object

OleQueryOutOfDate

Determines whether an object is out-of-date

OleSetLinkUpdateOptions

Sets link-update options for an object

OleUpdate

Updates the specified object

### Object-creation Functions (3.1)

<u>OleClone</u>	Makes a copy of an object
<u>OleCopyFromLink</u>	Makes an embedded copy of a linked object
<u>OleCreate</u>	Creates an object of a specified class
<u>OleCreateFromClip</u>	Creates an object from the clipboard
<u>OleCreateFromFile</u>	Creates an object from a file
<u>OleCreateFromTemplate</u>	Creates an object from a template
<u>OleCreateInvisible</u>	Creates an object without displaying it
<u>OleCreateLinkFromClip</u>	Creates a link to an object from the clipboard
<u>OleCreateLinkFromFile</u>	Creates a link to an object in a file
<u>OleLoadFromStream</u>	Loads an object from the containing document
<u>OleObjectConvert</u>	Creates a new object using a specified protocol
<u>OleQueryCreateFromClip</u>	Retrieves protocol data for clipboard object

### Object-management Functions (3.1)

<u>OleActivate</u>	Opens an object for an operation
<u>OleCopyToClipboard</u>	Puts the specified object on the clipboard
<u>OleDelete</u>	Deletes an object
<u>OleDraw</u>	Draws an object into a device context
<u>OleEnumFormats</u>	Enumerates data formats for an object
<u>OleEqual</u>	Compares two objects for equality
<u>OleGetData</u>	Retrieves data from an object in a specified format
<u>OleQueryBounds</u>	Retrieves the bounding rectangle for an object
<u>OleQueryClientVersion</u>	Retrieves the version number of a client library
<u>OleQueryName</u>	Retrieves the name of an object
<u>OleQueryProtocol</u>	Determines whether an object supports a protocol
<u>OleQuerySize</u>	Retrieves the size of an object
<u>OleQueryType</u>	Determines if object is linked, embedded, or static
<u>OleRelease</u>	Releases an object from memory
<u>OleSaveToStream</u>	Saves an object to the stream
<u>OleSetBounds</u>	Sets the bounding rectangle for an object
<u>OleSetColorScheme</u>	Specifies the client's recommended object colors
<u>OleSetData</u>	Sends data in the specified format to the server
<u>OleSetHostNames</u>	Sets the client name and object name for server
<u>OleSetTargetDevice</u>	Specifies the target device for an object

### Server Functions (Client) (3.1)

<u>OleClose</u>	Closes the specified open object
<u>OleExecute</u>	Sends DDE execute commands to a server
<u>OleLockServer</u>	Keeps an open server application in memory
<u>OleQueryOpen</u>	Determines whether an object is open
<u>OleQueryReleaseError</u>	Determines the status of a released operation
<u>OleQueryReleaseMethod</u>	Determines which operation released
<u>OleQueryReleaseStatus</u>	Determines whether an operation released
<u>OleReconnect</u>	Reconnects to an open linked object
<u>OleRequestData</u>	Retrieves data from a server in a specified format
<u>OleUnlockServer</u>	Releases a server locked with OleLockServer

### Server Functions (Server) (3.1)

<u>OleBlockServer</u>	Queues incoming requests for the server
<u>OleQueryServerVersion</u>	Retrieves the version number of the server library
<u>OleRegisterServer</u>	Registers the specified server
<u>OleRevokeObject</u>	Revokes access to an object
<u>OleRevokeServer</u>	Revokes the specified server
<u>OleUnblockServer</u>	Processes requests from a queue

### Alphabetized Windows functions (3.1)

<u>hread</u>	Reads from a file
<u>hwrite</u>	Writes to a file
<u>lclose</u>	Closes an open file
<u>lcreat</u>	Creates or opens a file
<u>llseek</u>	Repositions the file pointer
<u>lopen</u>	Opens a file
<u>lread</u>	Reads from a file
<u>lwrite</u>	Writes to a file
<u>AbortDoc</u>	Terminates a print job
<u>AccessResource</u>	Opens a resource file and locates a resource
<u>AddAtom</u>	Adds a string to the local atom table
<u>AddFontResource</u>	Adds a font to the font table
<u>AdjustWindowRect</u>	Computes the required size of a window rectangle
<u>AdjustWindowRectEx</u>	Computes the required size of a window rectangle
<u>AllocDiskSpace</u>	Creates a file to consume space on a disk partition.
<u>AllocDStoCSAlias</u>	Translates a data segment to a code segment
<u>AllocFileHandles</u>	Allocates up to 256 file handles
<u>AllocGDIMem</u>	Allocates all available memory in the <u>GDI</u> heap.
<u>AllocMem</u>	Allocates all available memory.
<u>AllocResource</u>	Allocates memory for a resource
<u>AllocSelector</u>	Allocates a new selector
<u>AllocUserMem</u>	Allocates all available memory in the User heap.
<u>AnimatePalette</u>	Replaces entries in a logical palette
<u>AnsiLower</u>	Converts a string to lower case
<u>AnsiLowerBuff</u>	Converts a string buffer to lower case
<u>AnsiNext</u>	Moves to the next character in a string
<u>AnsiPrev</u>	Move to the previous character in a string
<u>AnsiToOem</u>	Translates a Windows string to an OEM string
<u>AnsiToOemBuff</u>	Translates a Windows string to an OEM string
<u>AnsiUpper</u>	Converts a string to upper case
<u>AnsiUpperBuff</u>	Converts a string buffer to upper case
<u>AnyPopup</u>	Indicates if pop-up or overlapped window exists
<u>AppendMenu</u>	Appends a new item to a menu
<u>Arc</u>	Draws an arc
<u>ArrangeIconicWindows</u>	Arranges minimized child windows
<u>BeginDeferWindowPos</u>	Creates a window-position structure
<u>BeginPaint</u>	Prepares a window for painting
<u>BitBlt</u>	Copies a bitmap between device contexts
<u>BringWindowToTop</u>	Uncovers an overlapped window
<u>BuildCommDCB</u>	Translates a device definition string to a <u>DCB</u>
<u>CallMsgFilter</u>	Passes a message to a message-filter function
<u>CallNextHookEx</u>	Passes hook information down the hook chain
<u>CallWindowProc</u>	Passes a message to a window procedure
<u>Catch</u>	Captures the current execution environment
<u>ChangeClipboardChain</u>	Removes a window from the clipboard-viewer chain
<u>ChangeMenu</u>	Obsolete
<u>CheckDlgButton</u>	Changes a check mark by a dialog button
<u>CheckMenuItem</u>	Changes a check mark by a menu item
<u>CheckRadioButton</u>	Places a check mark by a radio button
<u>ChildWindowFromPoint</u>	Determines the window containing a point
<u>ChooseColor</u>	Creates a color-selection dialog box
<u>ChooseFont</u>	Creates a font-selection dialog box
<u>Chord</u>	Draws a chord
<u>ClassFirst</u>	Retrieves information about first class in class list
<u>ClassNext</u>	Retrieves information about next class in class list
<u>ClearCommBreak</u>	Restores character transmission
<u>ClientToScreen</u>	Converts client point to screen coordinates
<u>ClipCursor</u>	Confines the cursor to a specified rectangle
<u>CloseClipboard</u>	Closes the clipboard
<u>CloseComm</u>	Closes a communications device
<u>CloseDriver</u>	Closes an installable driver
<u>CloseMetaFile</u>	Closes metafile dc and gets handle
<u>CloseSound</u>	Obsolete

<u>CloseWindow</u>	Minimizes a window
<u>CombineRgn</u>	Creates a region by combining two regions
<u>CommDlgExtendedError</u>	Retrieves Error Data
<u>CopyCursor</u>	Copies a cursor
<u>CopyIcon</u>	Copies an icon
<u>CopyLZFile</u>	Copies a file and decompresses it if compressed
<u>CopyMetaFile</u>	Copies a metafile
<u>CopyRect</u>	Copies the dimensions of a rectangle
<u>CountClipboardFormats</u>	Returns the number of clipboard formats
<u>CountVoiceNotes</u>	Obsolete
<u>CreateBitmap</u>	Creates device-dependent memory bitmap
<u>CreateBitmapIndirect</u>	Creates bitmap using <u>BITMAP</u> structure
<u>CreateBrushIndirect</u>	Creates a brush with specified attributes
<u>CreateCaret</u>	Creates a new shape for the system caret
<u>CreateCompatibleBitmap</u>	Creates bitmap compatible with DC
<u>CreateCompatibleDC</u>	Creates a DC compatible with specified DC
<u>CreateCursor</u>	Creates a cursor with specified dimensions
<u>CreateDC</u>	Creates a device context
<u>CreateDialog</u>	Creates a modeless dialog box
<u>CreateDialogIndirect</u>	Creates modeless dialog box from memory template
<u>CreateDialogIndirectParam</u>	Creates modeless dialog box from memory template
<u>CreateDialogParam</u>	Creates a modeless dialog box
<u>CreateDIBitmap</u>	Creates bitmap handle from DIB spec
<u>CreateDIBPatternBrush</u>	Creates a pattern brush from a DIB
<u>CreateDiscardableBitmap</u>	Creates discardable bitmap
<u>CreateEllipticRgn</u>	Creates an elliptical region
<u>CreateEllipticRgnIndirect</u>	Creates an elliptical region
<u>CreateFont</u>	Creates a logical font
<u>CreateFontIndirect</u>	Creates a font using <u>LOGFONT</u> structure
<u>CreateHatchBrush</u>	Creates a hatched brush
<u>CreateIC</u>	Creates an information context
<u>CreateIcon</u>	Creates an icon with the specified dimensions
<u>CreateMenu</u>	Creates a menu
<u>CreateMetaFile</u>	Creates a metafile device context
<u>CreatePalette</u>	Creates a logical color palette
<u>CreatePatternBrush</u>	Creates a pattern brush from a bitmap
<u>CreatePen</u>	Creates a pen
<u>CreatePenIndirect</u>	Creates a pen using a <u>LOGPEN</u> structure
<u>CreatePolygonRgn</u>	Creates a polygonal region
<u>CreatePolyPolygonRgn</u>	Creates a region consisting of polygons
<u>CreatePopupMenu</u>	Creates a pop-up window
<u>CreateRectRgn</u>	Creates a rectangular region
<u>CreateRectRgnIndirect</u>	Creates a region using a <u>RECT</u> structure
<u>CreateRoundRectRgn</u>	Creates a rectangular region with round corners
<u>CreateScalableFontResource</u>	Creates resource file with font info
<u>CreateSolidBrush</u>	Creates a solid brush with a specified color
<u>CreateWindow</u>	Creates a window
<u>CreateWindowEx</u>	Creates a window
<u>DdeAbandonTransaction</u>	Abandons an asynchronous transaction
<u>DdeAccessData</u>	Accesses a DDE global memory object
<u>DdeAddData</u>	Adds data to a DDE global memory object
<u>DdeClientTransaction</u>	Begins a DDE data transaction
<u>DdeCmpStringHandles</u>	Compares two DDE string handles
<u>DdeConnect</u>	Establishes a conversation with a server
<u>DdeConnectList</u>	Establishes multiple DDE conversations
<u>DdeCreateDataHandle</u>	Creates a DDE data handle
<u>DdeCreateStringHandle</u>	Creates a DDE string handle
<u>DdeDisconnect</u>	Terminates a DDE conversation
<u>DdeDisconnectList</u>	Destroys a DDE conversation list
<u>DdeEnableCallback</u>	Enables or disables one or more DDE conversations
<u>DdeFreeDataHandle</u>	Frees a global memory object
<u>DdeFreeStringHandle</u>	Frees a DDE string handle
<u>DdeGetData</u>	Copies data from a global memory object to a buffer
<u>DdeGetLastError</u>	Returns an error code set by a DDEML function

<u>DdeInitialize</u>	Registers an application with the DDEML
<u>DdeKeepStringHandle</u>	Increments the usage count for a string handle
<u>DdeNameService</u>	Registers or unregisters a service name
<u>DdePostAdvise</u>	Prompts a server to send advise data to a client
<u>DdeQueryConvInfo</u>	Retrieves information about a DDE conversation
<u>DdeQueryNextServer</u>	Obtains the next handle in a conversation list
<u>DdeQueryString</u>	Copies string-handle text to a buffer
<u>DdeReconnect</u>	Reestablishes a conversation with a server
<u>DdeSetUserHandle</u>	Associates a user-defined handle with a transaction
<u>DdeUnaccessData</u>	Frees a DDE global memory object
<u>DdeUninitialize</u>	Frees an application's DDEML resources
<u>DebugBreak</u>	Causes a breakpoint exception
<u>DebugOutput</u>	sends messages to the debugging terminal
<u>DefDlgProc</u>	Provides default window message processing
<u>DefDriverProc</u>	Calls the default installable-driver procedure
<u>DeferWindowPos</u>	Updates a multiple window position structure
<u>DefFrameProc</u>	Default MDI frame window message processing
<u>DefHookProc</u>	Calls the next function in a hook-function chain
<u>DefMDIChildProc</u>	Default MDI child window message processing
<u>DefScreenSaverProc</u>	Calls default screen-saver window procedure
<u>DefWindowProc</u>	Calls the default window procedure
<u>DeleteAtom</u>	Decrements a local atom's reference count
<u>DeleteDC</u>	Deletes a device context
<u>DeleteMenu</u>	Deletes an item from a menu
<u>DeleteMetaFile</u>	Invalidates a metafile handle
<u>DeleteObject</u>	Deletes an object from memory
<u>DestroyCaret</u>	Destroys the current caret
<u>DestroyCursor</u>	Destroys a cursor
<u>DestroyIcon</u>	Destroys an icon
<u>DestroyMenu</u>	Destroys a menu
<u>DestroyWindow</u>	Destroys a window
<u>DeviceCapabilities</u>	Retrieves the capabilities of a device
<u>DeviceMode</u>	Displays dialog box for printing modes
<u>DialogBox</u>	Creates a modal dialog box
<u>DialogBoxIndirect</u>	Creates modal dialog box from template in memory
<u>DialogBoxIndirectParam</u>	Creates modal dialog box from template in memory
<u>DialogBoxParam</u>	Creates a modal dialog box
<u>DirectedYield</u>	Forces execution to continue at a specified task
<u>DispatchMessage</u>	Dispatches a message to a window
<u>DlgChangePassword</u>	Changes password for screen saver
<u>DlgDirList</u>	Fills a directory list box
<u>DlgDirListComboBox</u>	Fills a directory list box
<u>DlgDirSelect</u>	Retrieves a selection from a directory list box
<u>DlgDirSelectComboBox</u>	Retrieves a selection from a directory list box
<u>DlgDirSelectComboBoxEx</u>	Retrieves selection from dir. list box
<u>DlgDirSelectEx</u>	Retrieves a selection from a directory list box
<u>DlgGetPassword</u>	Retrieves password for screen saver
<u>DlgInvalidPassword</u>	Warns of invalid screen saver password
<u>DOS3Call</u>	Issues a DOS Int 21h function request
<u>DProtoP</u>	Converts device points to logical points
<u>DragAcceptFiles</u>	Registers whether a windows accepts dropped files
<u>DragFinish</u>	Releases memory allocated for dropping files
<u>DragQueryFile</u>	Retrieves filename of dropped file
<u>DragQueryPoint</u>	Retrieves mouse position at file drop
<u>DrawFocusRect</u>	Draws a rectangle in the focus style
<u>DrawIcon</u>	Draws an icon in the specified device context
<u>DrawMenuBar</u>	Redraws the menu bar
<u>DrawText</u>	Draws formatted text in a rectangle
<u>Ellipse</u>	Draws an ellipse
<u>EmptyClipboard</u>	Empties the clipboard and frees data handles
<u>EnableCommNotification</u>	Enables/disables <u>WM_COMMNOTIFY</u> posting to window
<u>EnableHardwareInput</u>	Controls mouse and keyboard input queuing
<u>EnableMenuItem</u>	Enables, disables, or grays a menu item
<u>EnableScrollBar</u>	Enables or disables scroll-bar arrows



<u>EnableWindow</u>	Sets the window-enable state
<u>EndDeferWindowPos</u>	Updates position and size of multiple windows
<u>EndDialog</u>	Hides a modal dialog box
<u>EndDoc</u>	Ends a print job
<u>EndPage</u>	Ends a page
<u>EndPaint</u>	Marks end of painting in the specified window
<u>EnumChildWindows</u>	Passes child-window handles to callback function
<u>EnumClipboardFormats</u>	Returns available clipboard formats
<u>EnumFontFamilies</u>	Retrieves fonts in a specified family
<u>EnumFonts</u>	Enumerates fonts on specified device
<u>EnumMetaFile</u>	Enumerates metafile records
<u>EnumObjects</u>	Enumerates pens and brushes in a device context
<u>EnumProps</u>	Passes property-list entries to callback
<u>EnumTaskWindows</u>	Passes task's window handles to callback
<u>EnumWindows</u>	Passes parent-window handles to a callback
<u>EqualRect</u>	Determines whether two rectangles are equal
<u>EqualRgn</u>	Compares two regions for equality
<u>Escape</u>	Allows access to device facilities
<u>EscapeCommFunction</u>	Passes an extended function to a device
<u>ExcludeClipRect</u>	Changes clipping region, excluding rectangle
<u>ExcludeUpdateRgn</u>	Excludes an updated region from clipping region
<u>ExitWindows</u>	Restarts or terminates Windows
<u>ExitWindowsExec</u>	Terminates Windows, runs MS-DOS app
<u>ExtDeviceMode</u>	Displays dialog box for printing modes
<u>ExtFloodFill</u>	Fills area with current brush
<u>ExtractIcon</u>	Retrieves handle of icon from executable file
<u>ExtTextOut</u>	Writes a character string in rectangular region
<u>FatalAppExit</u>	Terminates an application
<u>FatalExit</u>	Displays debug info., causes breakpoint exception
<u>FillRect</u>	Fills a rectangle with the specified brush
<u>FillRgn</u>	Fills a region with a brush
<u>FindAtom</u>	Retrieves a string atom from the local atom table
<u>FindExecutable</u>	Retrieves name and handle of program for a file
<u>FindResource</u>	Locates a resource in a resource file
<u>FindText</u>	Creates a find-text dialog box
<u>FindWindow</u>	Returns window handle for class and window name
<u>FlashWindow</u>	Flashes a window once
<u>FloodFill</u>	Fills area with current brush
<u>FlushComm</u>	Flushes a transmit or receive queue
<u>FrameRect</u>	Draws a window border with a specified brush
<u>FrameRgn</u>	Draws a border around a region
<u>FreeAllGDIMem</u>	Frees memory allocated by AllocGDIMem.
<u>FreeAllMem</u>	Frees memory allocated by AllocMem.
<u>FreeAllUserMem</u>	Frees memory allocated by AllocUserMem.
<u>FreeLibrary</u>	Unloads a library module instance
<u>FreeModule</u>	Unloads a module instance
<u>FreeProcInstance</u>	Frees a function instance
<u>FreeResource</u>	Unloads a resource instance
<u>FreeSelector</u>	Frees an allocated selector
<u>GetActiveWindow</u>	Retrieves the handle of the active window
<u>GetAspectRatioFilter</u>	Retrieves setting of aspect-ratio filter
<u>GetAspectRatioFilterEx</u>	Retrieves current aspect-ratio filter
<u>GetAsyncKeyState</u>	Determines key state
<u>GetAtomHandle</u>	Retrieves an atom handle
<u>GetAtomName</u>	Retrieves a local atom string
<u>GetBitmapBits</u>	Copies bitmap bits to a buffer
<u>GetBitmapDimension</u>	Retrieves width and height of bitmap
<u>GetBitmapDimensionEx</u>	Retrieves width and height of bitmap
<u>GetBkColor</u>	Retrieves the current background color
<u>GetBkMode</u>	Retrieves the background mode
<u>GetBoundsRect</u>	Returns current accumulated bounding rectangle
<u>GetBrushOrg</u>	Retrieves the origin of the current brush
<u>GetBrushOrgEx</u>	Retrieves the origin of the current brush
<u>GetCapture</u>	Returns the handle for the mouse-capture window

<u>GetCaretBlinkTime</u>	Returns the caret blink rate
<u>GetCaretPos</u>	Returns the current caret position
<u>GetCharABCWidths</u>	Retrieves widths of TrueType characters
<u>GetCharWidth</u>	Retrieves character widths
<u>GetClassInfo</u>	Returns window class information
<u>GetClassLong</u>	Returns window-class data
<u>GetClassName</u>	Returns window class name
<u>GetClassWord</u>	Returns window class memory word
<u>GetClientRect</u>	Returns window client area coordinates
<u>GetClipboardData</u>	Returns a handle to clipboard data
<u>GetClipboardFormatName</u>	Returns registered clipboard format name
<u>GetClipboardOwner</u>	Returns clipboard owner window handle
<u>GetClipboardViewer</u>	Returns first clipboard viewer window handle
<u>GetClipBox</u>	Retrieves rectangle for clipping region
<u>GetClipCursor</u>	Returns cursor-confining rectangle coordinates
<u>GetCodeHandle</u>	Determines the location of a function
<u>GetCodeInfo</u>	Retrieves code-segment information
<u>GetCommError</u>	Returns communications-device status
<u>GetCommEventMask</u>	Retrieves the device event mask
<u>GetCommState</u>	Reads communications device status
<u>GetCurrentPDB</u>	Returns the selector address of the current PDB
<u>GetCurrentPosition</u>	Retrieves current position
<u>GetCurrentPositionEx</u>	Retrieves position in logical units
<u>GetCurrentTask</u>	Returns current task handle
<u>GetCurrentTime</u>	Returns elapsed time since Windows started
<u>GetCursor</u>	Returns current cursor handle
<u>GetCursorPos</u>	Returns current cursor position
<u>GetDC</u>	Returns window device-context handle
<u>GetDCEx</u>	Retrieves the handle of a device context
<u>GetDCOrg</u>	Retrieves translation origin for device context
<u>GetDesktopWindow</u>	Returns desktop window handle
<u>GetDeviceCaps</u>	Retrieves device capabilities
<u>GetDialogBaseUnits</u>	Returns dialog base units
<u>GetDIBits</u>	Copies DIB bits into a buffer
<u>GetDlgCtrlID</u>	Returns child window ID
<u>GetDlgItem</u>	Returns handle of a dialog control
<u>GetDlgItemInt</u>	Translates dialog text into an integer
<u>GetDlgItemText</u>	Retrieves dialog control text
<u>GetDOSEnvironment</u>	Returns a far pointer to the current environment
<u>GetDoubleClickTime</u>	Returns mouse double click time
<u>GetDriverInfo</u>	Retrieves installable-driver data
<u>GetDriverModuleHandle</u>	Retrieves an installable-driver instance handle
<u>GetDriveType</u>	Determines drive type
<u>GetExpandedName</u>	Retrieves original filename for a compressed file
<u>GetFileResource</u>	Copies a resource into a buffer
<u>GetFileResourceSize</u>	Returns the size of a resource
<u>GetFileName</u>	Retrieves a filename
<u>GetFileVersionInfo</u>	Returns version information about a file
<u>GetFileVersionInfoSize</u>	Returns the size of a file's version information
<u>GetFocus</u>	Returns current focus window handle
<u>GetFontData</u>	Retrieves font metric data
<u>GetFreeFileHandles</u>	Returns the number of free file handles
<u>GetFreeSpace</u>	Returns number of free bytes in the global heap
<u>GetFreeSystemResources</u>	Returns percentage of free system resource space
<u>GetGlyphOutline</u>	Retrieves data for individual outline character
<u>GetInputState</u>	Returns mouse, keyboard and timer queue status
<u>GetInstanceData</u>	Copy previous instance data into current instance
<u>GetKBCodePage</u>	Returns the current code page
<u>GetKerningPairs</u>	Retrieves kerning pairs for current font
<u>GetKeyboardState</u>	Returns virtual-keyboard keys status
<u>GetKeyboardType</u>	Retrieves keyboard information
<u>GetKeyNameText</u>	Retrieves string representing the name of a key
<u>GetKeyState</u>	Returns specified virtual key state
<u>GetLastActivePopup</u>	Determines most recently active pop-up window

<u>GetMapMode</u>	Retrieves mapping mode
<u>GetMenu</u>	Returns menu handle for the specified window
<u>GetMenuCheckMarkDimensions</u>	Returns default check mark bitmap dimensions
<u>GetMenuItemCount</u>	Returns the number of items in a menu
<u>GetMenuItemID</u>	Returns a menu-item identifier
<u>GetMenuState</u>	Returns status flags for the specified menu item
<u>GetMenuString</u>	Copies a menu-item label into a buffer
<u>GetMessage</u>	Retrieves a message from the message queue
<u>GetMessageExtraInfo</u>	Retrieves information about a hardware message
<u>GetMessagePos</u>	Returns cursor position for last message
<u>GetMessageTime</u>	Returns the time for the last message
<u>GetMetaFile</u>	Creates handle to a metafile
<u>GetMetaFileBits</u>	Creates memory block from metafile
<u>GetModuleFileName</u>	Returns the file name for a module handle
<u>GetModuleHandle</u>	Returns a module handle for a named module
<u>GetModuleUsage</u>	Returns the reference count for a module
<u>GetNearestColor</u>	Retrieves closest available color
<u>GetNearestPaletteIndex</u>	Retrieves nearest match for a color
<u>GetNextDlgGroupItem</u>	Returns handle of previous or next group control
<u>GetNextDlgTabItem</u>	Returns the next or previous WS_TABSTOP control
<u>GetNextDriver</u>	Enumerates installable-driver instances
<u>GetNextWindow</u>	Returns next or previous window-manager window
<u>GetNumTasks</u>	Returns the current number of tasks
<u>GetObject</u>	Retrieves information about an object
<u>GetOpenClipboardWindow</u>	Returns handle to window that opened clipboard
<u>GetOpenFileName</u>	Creates an open-filename dialog box
<u>GetOutlineTextMetrics</u>	Retrieves metrics for TrueType fonts
<u>GetPaletteEntries</u>	Retrieves range of palette entries
<u>GetParent</u>	Returns parent window handle
<u>GetPixel</u>	Retrieves RGB color of specified pixel
<u>GetPolyFillMode</u>	Retrieves the current polygon-filling mode
<u>GetPriorityClipboardFormat</u>	Returns first clipboard format
<u>GetPrivateProfileInt</u>	Retrieves integer value from .ini file
<u>GetPrivateProfileString</u>	Retrieves a string from an initialization file
<u>GetProcAddress</u>	Returns the address of an exported DLL function
<u>GetProfileInt</u>	Retrieves an integer value from WIN.INI
<u>GetProfileString</u>	Retrieves a string from WIN.INI
<u>GetProp</u>	Returns data handle from a window property list
<u>GetQueueStatus</u>	Determines queued message type
<u>GetRasterizerCaps</u>	Retrieves status of TrueType on system
<u>GetRgnBox</u>	Retrieves bounding rectangle for region
<u>GetROP2</u>	Retrieves the current drawing mode
<u>GetSaveFileName</u>	Creates a save-filename dialog box
<u>GetScrollPos</u>	Returns current scroll-bar thumb position
<u>GetScrollRange</u>	Returns minimum and maximum scroll-bar positions
<u>GetSelectorBase</u>	Retrieves the base address of a selector
<u>GetSelectorLimit</u>	Retrieves the limit of a selector
<u>GetStockObject</u>	Retrieves handle of a stock pen, brush, or font
<u>GetStretchBltMode</u>	Retrieves the current bitmap-stretching mode
<u>GetSubMenu</u>	Returns pop-up menu handle
<u>GetSysColor</u>	Returns display-element color
<u>GetSysModalWindow</u>	Returns system-model window handle
<u>GetSystemDebugState</u>	Returns system-state information to a debugger
<u>GetSystemDir</u>	Returns the Windows system subdirectory
<u>GetSystemDirectory</u>	Returns the Windows system directory
<u>GetSystemMenu</u>	Provides access to the System menu
<u>GetSystemMetrics</u>	Retrieves the system metrics
<u>GetSystemPaletteEntries</u>	Retrieves entries from system palette
<u>GetSystemPaletteUse</u>	Determines use of entire system palette
<u>GetTabbedTextExtent</u>	Determines dimensions of tabbed string
<u>GetTempDrive</u>	Returns a disk drive letter for temporary files
<u>GetTempFileName</u>	Creates a temporary filename
<u>GetTextAlign</u>	Retrieves text-alignment flags
<u>GetTextCharacterExtra</u>	Retrieves intercharacter spacing

<u>GetTextColor</u>	Retrieves the current text color
<u>GetTextExtent</u>	Determines dimensions of a string
<u>GetTextExtentPoint</u>	Retrieves dimensions of string
<u>GetTextFace</u>	Retrieves typeface name of the current font
<u>GetTextMetrics</u>	Retrieves the metrics for the current font
<u>GetThresholdEvent</u>	Obsolete
<u>GetThresholdStatus</u>	Obsolete
<u>GetTickCount</u>	Returns amount of time Windows has been running
<u>GetTimerResolution</u>	Retrieves the timer resolution
<u>GetTopWindow</u>	Returns handle for top child of given window
<u>GetUpdateRect</u>	Returns window update region dimensions
<u>GetUpdateRgn</u>	Returns window update region
<u>GetVersion</u>	Returns the current Dos and Windows versions
<u>GetViewportExt</u>	Retrieves viewport extent
<u>GetViewportExtEx</u>	Retrieves viewport extent
<u>GetViewportOrg</u>	Retrieves viewport origin
<u>GetViewportOrgEx</u>	Retrieves viewport origin
<u>GetWinDebugInfo</u>	Queries current system-debugging information
<u>GetWindow</u>	Returns specified window handle
<u>GetWindowDC</u>	Returns window device context
<u>GetWindowExt</u>	Retrieves window extents
<u>GetWindowExtEx</u>	Retrieves window extents
<u>GetWindowLong</u>	Returns long value from extra window memory
<u>GetWindowOrg</u>	Retrieves window origin
<u>GetWindowOrgEx</u>	Retrieves window origin
<u>GetWindowPlacement</u>	Returns window show state and min/max position
<u>GetWindowRect</u>	Retrieves a window's coordinates
<u>GetWindowsDir</u>	Returns the Windows directory
<u>GetWindowsDirectory</u>	Returns the Windows directory
<u>GetWindowTask</u>	Returns the task associated with a window
<u>GetWindowText</u>	Copies window title-bar text to a buffer
<u>GetWindowTextLength</u>	Returns length of window title bar text
<u>GetWindowWord</u>	Returns a word value from extra window memory
<u>GetWinFlags</u>	Returns system configuration flags
<u>GetWinMem32Version</u>	Retrieves version of the 32-bit memory API
<u>Global16PointerAlloc</u>	Converts 16:32 pointer to 16:16
<u>Global16PointerFree</u>	Frees a 16:16 pointer alias
<u>Global32Alloc</u>	Allocates a USE32 memory object
<u>Global32CodeAlias</u>	Creates USE32 alias selector for 32-bit object
<u>Global32CodeAliasFree</u>	Frees a USE32 code-segment alias selector
<u>Global32Free</u>	Frees a USE32 memory object
<u>Global32Realloc</u>	Changes size of a USE32 memory object
<u>GlobalAddAtom</u>	Adds a string to the system atom table
<u>GlobalAlloc</u>	Allocates memory from the global heap
<u>GlobalCompact</u>	Generates free global memory by compacting
<u>GlobalDeleteAtom</u>	Decrements a global atom's reference count
<u>GlobalDosAlloc</u>	Allocates memory available to DOS in real mode
<u>GlobalDosFree</u>	Frees global memory allocated by GlobalDosAlloc
<u>GlobalEntryHandle</u>	Retrieves information about global memory object
<u>GlobalEntryModule</u>	Retrieves information about specific memory object
<u>GlobalFindAtom</u>	Retrieves string atom from the global atom table
<u>GlobalFirst</u>	Retrieves information about first global memory object
<u>GlobalFix</u>	Locks a global memory block in linear memory
<u>GlobalFlags</u>	Returns information about a global memory object
<u>GlobalFree</u>	Frees a global memory object
<u>GlobalGetAtomName</u>	Retrieves a global atom string
<u>GlobalHandle</u>	Returns a handle for a specified selector
<u>GlobalHandleToSel</u>	Converts a global handle to a selector
<u>GlobalInfo</u>	Retrieves information about the global heap
<u>GlobalLock</u>	Locks global memory object and returns a pointer
<u>GlobalLRUNewest</u>	Moves global memory block to newest LRU position
<u>GlobalLRUOldest</u>	Moves global memory block to oldest LRU position
<u>GlobalNext</u>	Retrieves information about next global memory object
<u>GlobalNotify</u>	Installs a notification procedure

<u>GlobalPageLock</u>	Increments global memory page-lock count
<u>GlobalPageUnlock</u>	Decrements global memory page-lock count
<u>GlobalReAlloc</u>	Changes size/attributes of global memory object
<u>GlobalSize</u>	Returns the size of a global memory object
<u>GlobalUnfix</u>	Unlocks a global-memory block in linear memory
<u>GlobalUnlock</u>	Unlocks a global memory object
<u>GlobalUnWire</u>	Should not be used
<u>GlobalWire</u>	Should not be used
<u>GrayString</u>	Draws gray text at the specified location
<u>HelpMessageFilterHookFunction</u>	Posts screen saver help message
<u>HideCaret</u>	Removes the caret from the screen
<u>HiLiteMenuItem</u>	Changes highlight of top-level menu item
<u>hmemcpy</u>	Copies bytes
<u>InflateRect</u>	Changes rectangle dimensions
<u>InitAtomTable</u>	Sets the size of the local atom table
<u>InSendMessage</u>	Determines if a window is processing SendMessage
<u>InsertMenu</u>	Inserts a new item in a menu
<u>InterruptRegister</u>	Installs function to handle system interrupts
<u>InterruptUnRegister</u>	Removes function that processed system interrupts
<u>IntersectClipRect</u>	Creates clipping region from intersection
<u>IntersectRect</u>	Calculates a rectangle intersection
<u>InvalidateRect</u>	Adds a rectangle to the update region
<u>InvalidateRgn</u>	Adds a region to the update region
<u>InvertRect</u>	Inverts a rectangular region
<u>InvertRgn</u>	Inverts the colors in a region
<u>IsBadCodePtr</u>	Determines whether a code pointer is valid
<u>IsBadHugeReadPtr</u>	Determines if a huge read pointer is valid
<u>IsBadHugeWritePtr</u>	Determines if a huge write pointer is valid
<u>IsBadReadPtr</u>	Determines whether a read pointer is valid
<u>IsBadStringPtr</u>	Determines whether a string pointer is valid
<u>IsBadWritePtr</u>	Determines whether a write pointer is valid
<u>IsCharAlpha</u>	Determines if a character is alphabetical
<u>IsCharAlphaNumeric</u>	Determines if a character is alphanumeric
<u>IsCharLower</u>	Determines if a character is lower case
<u>IsCharUpper</u>	Determines if a character is upper case
<u>IsChild</u>	Determines if a window is a child
<u>IsClipboardFormatAvailable</u>	Determines if specified format data is available
<u>IsDBCSLeadByte</u>	Determines if a character is a DBCS lead byte
<u>IsDialogMessage</u>	Determines if a message is for a dialog box
<u>IsDlgButtonChecked</u>	Determines the state of a button control
<u>IsGDIObject</u>	Determines if handle is not GDI object
<u>IsIconic</u>	Determines if a window is minimized
<u>IsMenu</u>	Determines if a menu handle is valid
<u>IsRectEmpty</u>	Determines whether rectangle is empty
<u>IsTask</u>	Determines whether a task handle is valid
<u>IsWindow</u>	Determines if a window handle is valid
<u>IsWindowEnabled</u>	Determines if a window accepts user input
<u>IsWindowVisible</u>	Determines visibility state of a window
<u>IsZoomed</u>	Determines if a window is maximized
<u>KillTimer</u>	Removes a timer
<u>LimitEmsPages</u>	Obsolete
<u>LineDDA</u>	Computes successive points in a line
<u>LineTo</u>	Draws a line from the current position
<u>LoadAccelerators</u>	Loads an accelerator table
<u>LoadBitmap</u>	Loads a bitmap resource
<u>LoadCursor</u>	Loads a cursor resource
<u>LoadIcon</u>	Loads an icon resource
<u>LoadLibrary</u>	Returns a handle to a library module
<u>LoadMenu</u>	Loads a menu resource
<u>LoadMenuIndirect</u>	Obtains a menu handle for a menu template
<u>LoadModule</u>	Loads and executes a program
<u>LoadResource</u>	Returns a handle to a resource
<u>LoadString</u>	Loads a string resource
<u>LocalAlloc</u>	Allocate memory from the local heap

<u>LocalCompact</u>	Generates free local memory by compacting
<u>LocalFirst</u>	Retrieves information about first local memory object
<u>LocalFlags</u>	Returns local memory object information
<u>LocalFree</u>	Frees a local memory object
<u>LocalHandle</u>	Returns the handle of a local memory object
<u>LocalInfo</u>	Fills structure with information about local heap
<u>LocalInit</u>	Initializes a local heap
<u>LocalLock</u>	Locks a local memory object and returns a pointer
<u>LocalNext</u>	Retrieves information about next local memory object
<u>LocalReAlloc</u>	Changes local memory size or attributes
<u>LocalShrink</u>	Shrinks the specified local heap
<u>LocalSize</u>	Returns the size of a local memory object
<u>LocalUnlock</u>	Unlocks a local memory object
<u>LockInput</u>	Locks input to all tasks except the current one
<u>LockResource</u>	Returns the address of a resource
<u>LockSegment</u>	Locks a discardable memory segment
<u>LockWindowUpdate</u>	Disables or reenables drawing in a window
<u>LogError</u>	Identifies an error message
<u>LogParamError</u>	Identifies a parameter validation error
<u>LProDP</u>	Converts logical points to device points
<u>lstrcat</u>	Appends one string to another
<u>lstrcmp</u>	Compares two character strings
<u>lstrcmpi</u>	Compares two character strings
<u>lstrcpy</u>	Copies a string to a buffer
<u>lstrlen</u>	Returns the number of characters in a string
<u>LZClose</u>	Closes a file
<u>LZCopy</u>	Copies a file and expands it if compressed
<u>LZDone</u>	Frees buffers allocated by LZStart
<u>LZInit</u>	Initializes data structures needed for decompression
<u>LZOpenFile</u>	Opens a file (both compressed and uncompressed)
<u>LZRead</u>	Reads a specified number of bytes from a compressed file
<u>LZSeek</u>	Repositions pointer in file
<u>LZStart</u>	Allocates buffers for CopyLZFile function
<u>MakeProcInstance</u>	Returns the address of prolog code for a function
<u>MapDialogRect</u>	Maps dialog box units to pixels
<u>MapVirtualKey</u>	Translates a virtual-key code or scan code
<u>MapWindowPoints</u>	Converts points to another coordinate system
<u>MemManInfo</u>	Retrieves information about the memory manager
<u>MemoryRead</u>	Reads memory from an arbitrary global heap object
<u>MemoryWrite</u>	Writes memory to an arbitrary global heap object
<u>MessageBeep</u>	Generates a beep
<u>MessageBox</u>	Creates a message-box window
<u>ModifyMenu</u>	Changes an existing menu item
<u>ModuleFindHandle</u>	Retrieves information about a module
<u>ModuleFindName</u>	Retrieves information about a module
<u>ModuleFirst</u>	Retrieves information about first module
<u>ModuleNext</u>	Retrieves information about next module
<u>MoveTo</u>	Moves the current position
<u>MoveToEx</u>	Moves the current position
<u>MoveWindow</u>	Changes the position and dimensions of a window
<u>MulDiv</u>	Multiplies two values and divides the result
<u>NetBIOSCall</u>	Issues a NETBIOS Interrupt 5Ch
<u>NotifyRegister</u>	Installs a notification callback function
<u>NotifyUnRegister</u>	Removes a notification callback function
<u>OemKeyScan</u>	Maps OEM ASCII to scan codes
<u>OemToAnsi</u>	Translates an OEM string to a Windows string
<u>OemToAnsiBuff</u>	Translates an OEM string to a Windows string
<u>OffsetClipRgn</u>	Moves a clipping region
<u>OffsetRect</u>	Moves a rectangle by an offset
<u>OffsetRgn</u>	Moves a region by a specified offset
<u>OffsetViewportOrg</u>	Moves viewport origin
<u>OffsetViewportOrgEx</u>	Moves viewport origin
<u>OffsetWindowOrg</u>	Moves window origin
<u>OffsetWindowOrgEx</u>	Moves window origin

<u>OleActivate</u>	Activates an object
<u>OleBlockServer</u>	Queues incoming requests for the server
<u>OleClone</u>	Makes a copy of an object
<u>OleClose</u>	Closes specified object
<u>OleCopyFromLink</u>	Makes an embedded copy of a linked object
<u>OleCopyToClipboard</u>	Puts the specified object on the clipboard
<u>OleCreate</u>	Creates an object of a specified class
<u>OleCreateFromClip</u>	Creates an object from the clipboard
<u>OleCreateFromFile</u>	Creates an object from a file
<u>OleCreateFromTemplate</u>	Creates an object from a template
<u>OleCreateInvisible</u>	Creates an object without displaying it
<u>OleCreateLinkFromClip</u>	Creates link to object from the clipboard
<u>OleCreateLinkFromFile</u>	Creates link to object in a file
<u>OleDelete</u>	Deletes an object
<u>OleDraw</u>	Draws an object into a device context
<u>OleEnumFormats</u>	Enumerates data formats for an object
<u>OleEnumObjects</u>	Enumerates objects in a document
<u>OleEqual</u>	Compares two objects for equality
<u>OleExecute</u>	Sends DDE execute commands to a server
<u>OleGetData</u>	Retrieves data for an object in a specified format
<u>OleGetLinkUpdateOptions</u>	Retrieves update options for an object
<u>OleIsDcMeta</u>	Identifies metafile device context
<u>OleLoadFromStream</u>	Loads an object from containing document
<u>OleLockServer</u>	Keeps server in memory
<u>OleObjectConvert</u>	Creates a new object using a specified protocol
<u>OleQueryBounds</u>	Retrieves bounding rectangle for object
<u>OleQueryClientVersion</u>	Retrieves version of client library
<u>OleQueryCreateFromClip</u>	Retrieves create data for clipboard object
<u>OleQueryLinkFromClip</u>	Retrieves link data for clipboard object
<u>OleQueryName</u>	Retrieves the name of an object
<u>OleQueryOpen</u>	Determines whether an object is open
<u>OleQueryOutOfDate</u>	Determines whether an object is out-of-date
<u>OleQueryProtocol</u>	Determines if an object supports a protocol
<u>OleQueryReleaseError</u>	Determines status of released operation
<u>OleQueryReleaseMethod</u>	Determines which operation released
<u>OleQueryReleaseStatus</u>	Determines whether an operation released
<u>OleQueryServerVersion</u>	Retrieves version of server library
<u>OleQuerySize</u>	Retrieves the size of an object
<u>OleQueryType</u>	Determines if object is linked, embedded, or static
<u>OleReconnect</u>	Reconnects to an open linked object
<u>OleRegisterClientDoc</u>	Registers a document with the library
<u>OleRegisterServer</u>	Registers the specified server
<u>OleRegisterServerDoc</u>	Registers document with server library
<u>OleRelease</u>	Releases an object from memory
<u>OleRename</u>	Informs library an object is renamed
<u>OleRenameClientDoc</u>	Informs library a document is renamed
<u>OleRenameServerDoc</u>	Informs library a document is renamed
<u>OleRequestData</u>	Retrieves data from a server in a specified format
<u>OleRevertClientDoc</u>	Informs library a doc reverted to saved state
<u>OleRevertServerDoc</u>	Informs library a doc is reset to saved state
<u>OleRevokeClientDoc</u>	Informs library a document is not open
<u>OleRevokeObject</u>	Revokes access to an object
<u>OleRevokeServer</u>	Revokes the specified server
<u>OleRevokeServerDoc</u>	Revokes the specified document
<u>OleSavedClientDoc</u>	Informs library a doc has been saved
<u>OleSavedServerDoc</u>	Informs library a doc has been saved
<u>OleSaveToStream</u>	Saves an object to the stream
<u>OleSetBounds</u>	Sets bounding rectangle for object
<u>OleSetColorScheme</u>	Specifies client's recommended object colors
<u>OleSetData</u>	Sends data in specified format to server
<u>OleSetHostNames</u>	Sets client name and object name for server
<u>OleSetLinkUpdateOptions</u>	Sets update options for an object
<u>OleSetTargetDevice</u>	Sets target device for an object
<u>OleUnblockServer</u>	Processes requests from queue

<u>OleUnlockServer</u>	Releases server locked with OleLockServer
<u>OleUpdate</u>	Updates an object
<u>OpenClipboard</u>	Opens the clipboard
<u>OpenComm</u>	Opens a communications device
<u>OpenDriver</u>	Opens an installable driver
<u>OpenFile</u>	Creates, opens, reopens or deletes a file
<u>OpenIcon</u>	Activates a minimized window
<u>OpenSound</u>	Obsolete
<u>OutputDebugString</u>	Sends a character string to the debugger
<u>PaintRgn</u>	Fills region with brush in device context
<u>PatBlt</u>	Creates a bitmap pattern
<u>PeekMessage</u>	Checks message queue
<u>Pie</u>	Draws a pie-shaped wedge
<u>PlayMetaFile</u>	Plays a metafile
<u>PlayMetaFileRecord</u>	Plays a metafile record
<u>Polygon</u>	Draws a polygon
<u>Polyline</u>	Draws line segments to connect specified points
<u>PolyPolygon</u>	Draws a series of polygons
<u>PostAppMessage</u>	Posts a message to an application
<u>PostMessage</u>	Places a message in a window's message queue
<u>PostQuitMessage</u>	Tells Windows that an application is terminating
<u>PrestoChangeSelector</u>	Generates code selector from data
<u>PrintDlg</u>	Creates a print-text dialog box
<u>ProfClear</u>	Discards all buffered Profiler samples
<u>ProfEInish</u>	Stops profile sampling and flushes profile buffer
<u>ProfFlush</u>	Flushes the Profiler sampling buffer to disk
<u>ProfInsChk</u>	Determines whether Profiler is installed
<u>ProfSampRate</u>	Sets the Profiler sampling rate
<u>ProfSetup</u>	Sets Profiler buffer size and sample quantity
<u>ProfStart</u>	Starts profile sampling
<u>ProfStop</u>	Stops profile sampling
<u>PtInRect</u>	Determines if a point is in a rectangle
<u>PtInRegion</u>	Queries whether a point is in a region
<u>PtVisible</u>	Queries whether point is within clipping region
<u>QueryAbort</u>	Queries whether to terminate a print job
<u>QuerySendMessage</u>	Determines if a message originated within a task
<u>ReadComm</u>	Reads from a communications device
<u>RealizePalette</u>	Maps entries from logical to system palette
<u>Rectangle</u>	Draws a rectangle
<u>RectInRegion</u>	Queries whether rectangle overlaps region
<u>RectVisible</u>	Queries whether rectangle is in clip region
<u>RedrawWindow</u>	Updates a client rectangle or region
<u>RegCloseKey</u>	Closes a key
<u>RegCreateKey</u>	Creates a key
<u>RegDeleteKey</u>	Deletes a key
<u>RegEnumKey</u>	Enumerates subkeys of specified key
<u>RegisterClass</u>	Registers a window class
<u>RegisterClipboardFormat</u>	Registers a new clipboard format
<u>RegisterDialogClasses</u>	Registers dialog classes for screen-savers
<u>RegisterWindowMessage</u>	Defines a new unique window message
<u>RegOpenKey</u>	Opens a key
<u>RegQueryValue</u>	Retrieves text string for specified key
<u>RegSetValue</u>	Associates a text string with a specified key
<u>ReleaseCapture</u>	Releases mouse capture
<u>ReleaseDC</u>	Frees a device context
<u>RemoveFontResource</u>	Removes font resource
<u>RemoveMenu</u>	Deletes a menu item and pop-up menu
<u>RemoveProp</u>	Removes a property-list entry
<u>ReplaceText</u>	Creates a replace-text dialog box
<u>ReplyMessage</u>	Replies to a SendMessage
<u>ResetDC</u>	Updates a device context
<u>ResizePalette</u>	Changes the size of a logical palette
<u>RestoreDC</u>	Restores device context
<u>RoundRect</u>	Draws a rectangle with rounded corners



<u>SaveDC</u>	Saves current state of device context
<u>ScaleViewportExt</u>	Scales viewport extents
<u>ScaleViewportExtEx</u>	Scales viewport extents
<u>ScaleWindowExt</u>	Scales window extents
<u>ScaleWindowExtEx</u>	Scales window extents
<u>ScreenSaverConfigureDialog</u>	Processes input to screensaver config. dialog
<u>ScreenSaverProc</u>	Processes input to a screen-saver window
<u>ScreenToClient</u>	Converts screen point to client coordinates
<u>ScrollDC</u>	Scrolls a rectangle horizontally and vertically
<u>ScrollWindow</u>	Scrolls a window's client area
<u>ScrollWindowEx</u>	Scrolls a window's client area
<u>SelectClipRgn</u>	Selects clipping region for device context
<u>SelectObject</u>	Selects object into a device context
<u>SelectPalette</u>	Selects a palette into a device context
<u>SendDlgItemMessage</u>	Sends a message to a dialog box control
<u>SendDriverMessage</u>	Sends a message to an installable driver
<u>SendMessage</u>	Sends a message to a window
<u>SetAbortProc</u>	Sets the abort function for a print job
<u>SetActiveWindow</u>	Makes a top-level window active
<u>SetBitmapBits</u>	Sets bitmap bits from array of bytes
<u>SetBitmapDimension</u>	Sets width and height of bitmap
<u>SetBitmapDimensionEx</u>	Sets width and height of bitmap
<u>SetBkColor</u>	Sets the current background color
<u>SetBkMode</u>	Sets the background mode
<u>SetBoundsRect</u>	Controls bounding-rectangle accumulation
<u>SetBrushOrg</u>	Sets the origin of the current brush
<u>SetCapture</u>	Sets the mouse capture to a window
<u>SetCaretBlinkTime</u>	Sets caret blink rate
<u>SetCaretPos</u>	Sets the caret position
<u>SetClassLong</u>	Sets a long value in extra class memory
<u>SetClassWord</u>	Sets a word value in extra class memory
<u>SetClipboardData</u>	Sets the data in the clipboard
<u>SetClipboardViewer</u>	Adds a window to the clipboard-viewer chain
<u>SetCommBreak</u>	Suspends character transmission
<u>SetCommEventMask</u>	Enables events in a device event mask
<u>SetCommState</u>	Sets communications-device state
<u>SetCursor</u>	Changes the mouse cursor
<u>SetCursorPos</u>	Sets mouse-cursor position in screen coordinates
<u>SetDIBits</u>	Sets the bits of a bitmap
<u>SetDIBitsToDevice</u>	Sets DIB bits to device
<u>SetDlgItemInt</u>	Converts an integer to a dialog text string
<u>SetDlgItemText</u>	Sets dialog title or item text
<u>SetDoubleClickTime</u>	Sets the mouse double-click time
<u>SetErrorMode</u>	Controls Interrupt 24h Error Handling
<u>SetFocus</u>	Sets the input focus to a window
<u>SetHandleCount</u>	Changes the number of available file handles
<u>SetKeyboardState</u>	Sets the keyboard state table
<u>SetMapMode</u>	Sets mapping mode
<u>SetMapperFlags</u>	Sets font-mapper flag
<u>SetMenu</u>	Sets the menu for a window
<u>SetMenuItemBitmaps</u>	Associates bitmaps with a menu item
<u>SetMessageQueue</u>	Creates a new message queue
<u>SetMetaFileBits</u>	Creates memory block from metafile
<u>SetMetaFileBitsBetter</u>	Creates memory block from metafile
<u>SetPaletteEntries</u>	Sets colors and flags for a color palette
<u>SetParent</u>	Changes a child's paren window
<u>SetPixel</u>	Sets pixel to specified color
<u>SetPolyFillMode</u>	Sets the polygon-filling mode
<u>SetProp</u>	Adds or changes a property-list entry
<u>SetRect</u>	Sets a rectangle's dimensions
<u>SetRectEmpty</u>	Creates an empty rectangle
<u>SetRectRgn</u>	Changes a region into specified rectangle
<u>SetResourceHandler</u>	Installs a load-resource callback function
<u>SetROP2</u>	Sets the current drawing mode

<u>SetScrollPos</u>	Sets scroll-bar thumb position
<u>SetScrollRange</u>	Sets minimum and maximum scroll-bar positions
<u>SetSelectorBase</u>	Sets the base and limit of a selector
<u>SetSelectorLimit</u>	Sets the limit of a selector
<u>SetSoundNoise</u>	Obsolete
<u>SetStretchBltMode</u>	Sets the bitmap-stretching mode
<u>SetSwapAreaSize</u>	Sets the amount of memory used for code segments
<u>SetSysColors</u>	Sets one or more system colors
<u>SetSysModalWindow</u>	Makes a window the system-modal window
<u>SetSystemPaletteUse</u>	Use of system palette static colors
<u>SetTextAlign</u>	Sets text-alignment flags
<u>SetTextCharacterExtra</u>	Sets intercharacter spacing
<u>SetTextColor</u>	Sets the foreground color of text
<u>SetTextJustification</u>	Sets alignment for text output
<u>SetTimer</u>	Installs a system timer
<u>SetViewportExt</u>	Sets viewport extents
<u>SetViewportExtEx</u>	Sets viewport extents
<u>SetViewportOrg</u>	Sets viewport origin
<u>SetViewportOrgEx</u>	Sets viewport origin
<u>SetVoiceAccent</u>	Obsolete
<u>SetVoiceEnvelope</u>	Obsolete
<u>SetVoiceNote</u>	Obsolete
<u>SetVoiceQueueSize</u>	Obsolete
<u>SetVoiceSound</u>	Obsolete
<u>SetVoiceThreshold</u>	Obsolete
<u>SetWinDebugInfo</u>	Sets current system-debugging information
<u>SetWindowExt</u>	Sets window extents
<u>SetWindowExtEx</u>	Sets window extents
<u>SetWindowLong</u>	Sets a long value in extra window memory
<u>SetWindowOrg</u>	Sets the window origin
<u>SetWindowOrgEx</u>	Sets the window origin
<u>SetWindowPlacement</u>	Sets window show state and min/max position
<u>SetWindowPos</u>	Sets a windows size, position, and order
<u>SetWindowsHook</u>	Installs a hook function
<u>SetWindowsHookEx</u>	Installs a hook function
<u>SetWindowText</u>	Sets text in a caption title or control window
<u>SetWindowWord</u>	Sets a word value in extra window memory
<u>ShellExecute</u>	Opens or prints specified file
<u>ShowCaret</u>	Shows (unhides) the caret
<u>ShowCursor</u>	Shows or hides the mouse cursor
<u>ShowOwnedPopups</u>	Shows or hides pop-up windows
<u>ShowScrollBar</u>	Shows or hides a scroll bar
<u>ShowWindow</u>	Sets window visibility state
<u>SizeofResource</u>	Returns the size of a resource
<u>SpoolFile</u>	Puts a file in the spooler queue
<u>StackTraceCSIPFirst</u>	Retrieves information about a stack frame
<u>StackTraceFirst</u>	Retrieves information about the first stack frame
<u>StackTraceNext</u>	Retrieves information about the next stack frame
<u>StartDoc</u>	Starts a print job
<u>StartPage</u>	Prepares printer driver to receive data
<u>StartSound</u>	Obsolete
<u>StopSound</u>	Obsolete
<u>StretchBlt</u>	Copies a bitmap, transforming if required
<u>StretchDIBits</u>	Moves DIB from source to destination rectangle
<u>SubtractRect</u>	Creates rect from difference of two rects
<u>SwapMouseButton</u>	Reverses the meaning of the mouse buttons
<u>SwapRecording</u>	Starts or stops memory swap recording
<u>SwitchStackBack</u>	Restores the current-task stack
<u>SwitchStackTo</u>	Changes the location of the stack
<u>SyncAllVoices</u>	Obsolete
<u>SystemHeapInfo</u>	Retrieves information about the <u>USER</u> heap
<u>SystemParametersInfo</u>	Queries or sets systemwide parameters
<u>TabbedTextOut</u>	Writes a tabbed character string
<u>TaskFindHandle</u>	Retrieves information about a task

<u>TaskFirst</u>	Retrieves information about first task in task queue
<u>TaskGetCSIP</u>	Returns the next CS:IP value of a task.
<u>TaskNext</u>	Retrieves information about next task in the task queue
<u>TaskSetCSIP</u>	Sets the CS:IP of a sleeping task.
<u>TaskSwitch</u>	Switches to a specific address within a new task
<u>TerminateApp</u>	Terminates an application
<u>TextOut</u>	Writes a character string at specified location
<u>Throw</u>	Restores the execution environment
<u>TimerCount</u>	Retrieves execution times
<u>ToAscii</u>	Translates virtual-key code to Windows character
<u>TrackPopupMenu</u>	Displays and tracks a pop-up menu
<u>TranslateAccelerator</u>	Processes menu command keyboard accelerators
<u>TranslateMDISysAccel</u>	Processes MDI keyboard accelerators
<u>TranslateMessage</u>	Translates virtual-key messages
<u>TransmitCommChar</u>	Places a character in the transmit queue
<u>UnAllocDiskSpace</u>	Deletes the file created by AllocDiskSpace.
<u>UnAllocFileHandles</u>	Frees file handles allocated by AllocFileHandles.
<u>UngetCommChar</u>	Puts a character back in the receive queue
<u>UnhookWindowsHook</u>	Removes a filter function
<u>UnhookWindowsHookEx</u>	Removes a function from the hook chain
<u>UnionRect</u>	Creates the union of two rectangles
<u>UnlockSegment</u>	Unlocks a discardable memory segment
<u>UnrealizeObject</u>	Resets brush origins and realizes palettes
<u>UnregisterClass</u>	Removes a window class
<u>UpdateColors</u>	Updates colors in client area
<u>UpdateWindow</u>	Updates a window's client area
<u>ValidateCodeSegments</u>	Test for memory overwrites
<u>ValidateFreeSpaces</u>	Checks free memory for valid contents
<u>ValidateRect</u>	Removes a rectangle from the update region
<u>ValidateRgn</u>	Removes a region from the update region
<u>VerFindFile</u>	Determines where to install a file
<u>VerInstallFile</u>	Installs a file
<u>VerLanguageName</u>	Converts a binary language identifier into a string
<u>VerQueryValue</u>	Returns version information about a block
<u>VkKeyScan</u>	Translates Windows character to virtual-key code
<u>WaitMessage</u>	Suspends an application and yields control
<u>WaitSoundState</u>	Obsolete
<u>WindowFromPoint</u>	Returns window containing a point
<u>WinExec</u>	Runs a program
<u>WinHelp</u>	Invokes Windows Help
<u>WNetAddConnection</u>	Adds network connections
<u>WNetCancelConnection</u>	Removes network connections
<u>WNetGetConnection</u>	Lists network connections
<u>WriteComm</u>	Writes to a communications device
<u>WritePrivateProfileString</u>	Writes a string to an initialization file
<u>WriteProfileString</u>	Writes a string to WIN.INI
<u>wsprintf</u>	Formats a string
<u>wvsprintf</u>	Formats a string
<u>XTYP_ADVDATA</u>	Passes advise data to a client
<u>XTYP_ADVREQ</u>	Prompts a server to send advise data to a client
<u>XTYP_ADVSTART</u>	Requests an advise loop
<u>XTYP_ADVSTOP</u>	Ends an advise loop
<u>XTYP_CONNECT</u>	Requests a DDE conversation
<u>XTYP_CONNECT_CONFIRM</u>	Confirms a DDE conversation
<u>XTYP_DISCONNECT</u>	Terminates a DDE conversation
<u>XTYP_ERROR</u>	Notifies a DDEML application of a critical error
<u>XTYP_EXECUTE</u>	Executes a server command
<u>XTYP_MONITOR</u>	Informs a DDE monitor application of a DDE event
<u>XTYP_POKE</u>	Sends unsolicited data to a server
<u>XTYP_REGISTER</u>	Registers a service name
<u>XTYP_REQUEST</u>	Requests data from a server
<u>XTYP_UNREGISTER</u>	Unregisters a service name
<u>XTYP_WILDCONNECT</u>	Requests multiple DDE conversation
<u>XTYP_XACT_COMPLETE</u>	Confirms completion of asynchronous transaction

Yield

Stops the current task

## Message groups (3.1)

Button messages

Clipboard messages

Combo box messages

DDE messages

Edit-control messages

Installable-driver messages

List box messages

## Button messages

<u>BM_GETCHECK</u>	Retrieves the button check state
<u>BM_GETSTATE</u>	Determines the state of a button or check box
<u>BM_SETCHECK</u>	Sets the button check state
<u>BM_SETSTATE</u>	Sets the highlighting state of a button
<u>BM_SETSTYLE</u>	Sets the style of a button

## Clipboard messages

<u>WM_ASKCBFORMATNAME</u>	Retrieves the name of the clipboard format
<u>WM_CHANGECHAIN</u>	Notifies clipboard viewer of removal from chain
<u>WM_COPY</u>	Copies a selection to the clipboard
<u>WM_CUT</u>	Deletes a selection and copies it to the clipboard
<u>WM_DESTROYCLIPBOARD</u>	Notifies owner that the clipboard was emptied
<u>WM_DRAWCLIPBOARD</u>	Indicates the clipboard's contents have changed
<u>WM_HSCROLLCLIPBOARD</u>	Prompts owner to scroll clipboard contents
<u>WM_PAINTCLIPBOARD</u>	Prompts owner to display clipboard contents
<u>WM_PASTE</u>	Inserts clipboard data into an edit control
<u>WM_RENDERALLFORMATS</u>	Notifies owner to render all clipboard formats
<u>WM_RENDERFORMAT</u>	Notifies owner to render clipboard data
<u>WM_SIZECLIPBOARD</u>	Indicates a change in the clipboard's size
<u>WM_VSCROLLCLIPBOARD</u>	Prompts owner to scroll clipboard contents

## Combo box messages

<u>CB_ADDSTRING</u>	Adds a string to the list box of a combo box
<u>CB_DELETESTRING</u>	Deletes list-box string in a combo box
<u>CB_DIR</u>	Adds file names to the list box of a combo box
<u>CB_FINDSTRING</u>	Finds a string in the list box of a combo box
<u>CB_GETCOUNT</u>	Gets the number of list-box items in a combo box
<u>CB_GETCURSEL</u>	Gets index of selected list-box item in combo box
<u>CB_GETDROPPEDCONTROLRECT</u>	Gets rectangle of combo-box drop-down list box
<u>CB_GETDROPPEDSTATE</u>	Determines if a combo box's list box is visible
<u>CB_GETEDITSEL</u>	Gets position of selection in edit control
<u>CB_GETEXTENDEDUI</u>	Determines if combo box has extended interface
<u>CB_GETITEMDATA</u>	Retrieves a value associated with an item
<u>CB_GETITEMHEIGHT</u>	Retrieves the height of items in a combo box
<u>CB_GETLBTEXT</u>	Gets a string from the list box of a combo box
<u>CB_GETLBTEXTLEN</u>	Gets length of a list-box string in a combo box
<u>CB_INSERTSTRING</u>	Inserts a string into the list box of a combo box
<u>CB_LIMITTEXT</u>	Limits amount of edit-control text in a combo box
<u>CB_RESETCONTENT</u>	Removes all items from list box of a combo box
<u>CB_SELECTSTRING</u>	Selects a string in the list box of a combo box
<u>CB_SETCURSEL</u>	Selects a string in the list box of a combo box
<u>CB_SETEDITSEL</u>	Sets the edit-control selection of a combo box
<u>CB_SETEXTENDEDUI</u>	Sets the default or extended user interface
<u>CB_SETITEMDATA</u>	Sets the value associated with an item
<u>CB_SETITEMHEIGHT</u>	Sets the height of items in a combo box
<u>CB_SHOWDROPDOWN</u>	Shows or hides the list box of a combo box



DDE messages

<u>WM_DDE_ACK</u>	Acknowledges the receipt of a DDE transaction
<u>WM_DDE_ADVISE</u>	Starts an advise loop with a DDE server
<u>WM_DDE_DATA</u>	Passes a data item to a DDE client
<u>WM_DDE_EXECUTE</u>	Passes a command to a DDE server
<u>WM_DDE_INITIATE</u>	Initiates a DDE conversation
<u>WM_DDE_POKE</u>	Send unsolicited data to a server
<u>WM_DDE_REQUEST</u>	Requests a data item from a DDE server
<u>WM_DDE_TERMINATE</u>	Terminates a DDE conversation
<u>WM_DDE_UNADVISE</u>	Ends a DDE advise loop

Edit-control messages

<u>EM_CANUNDO</u>	Determines if edit-control operation can be undone
<u>EM_EMPTYUNDOBUFFER</u>	Resets (clears) the edit-control undo flag
<u>EM_FMTLINES</u>	Sets soft line-break characters on or off
<u>EM_GETFIRSTVISIBLELINE</u>	Gets index of top line in an edit control
<u>EM_GETHANDLE</u>	Gets the handle of the memory for an MLE
<u>EM_GETLINE</u>	Retrieves a line from an MLE
<u>EM_GETLINECOUNT</u>	Retrieves number of lines in an MLE
<u>EM_GETMODIFY</u>	Checks whether edit-control contents have changed
<u>EM_GETRECT</u>	Gets the coordinates of an edit-control rectangle
<u>EM_GETSEL</u>	Gets position of current edit-control selection
<u>EM_LIMITTEXT</u>	Limits amount of text in an edit control
<u>EM_LINEFROMCHAR</u>	Retrieves a line number from a character index
<u>EM_LINEINDEX</u>	Retrieves the character index of an MLE line
<u>EM_LINELENGTH</u>	Retrieves the length of a line in an MLE
<u>EM_LINESCROLL</u>	Scrolls text in an MLE
<u>EM_REPLACESEL</u>	Replaces current selection in an edit control
<u>EM_SETHANDLE</u>	Sets the memory handle for an MLE
<u>EM_SETMODIFY</u>	Sets or clears edit-control modification flag
<u>EM_SETPASSWORDCHAR</u>	Sets or removes edit-control password character
<u>EM_SETREADONLY</u>	Sets the read-only state of an edit control
<u>EM_SETRECT</u>	Sets the formatting rectangle of an MLE
<u>EM_SETRECTNP</u>	Sets the formatting rectangle of an MLE
<u>EM_SETSEL</u>	Selects text within an edit control
<u>EM_SETTABSTOPS</u>	Sets the tab stops in an MLE
<u>EM_SETWORDBREAKPROC</u>	Provides custom word breaks in edit controls
<u>EM_UNDO</u>	Undoes the last operation in an edit control

List box messages

<u>LB_ADDSTRING</u>	Adds a string to a list box
<u>LB_DELETESTRING</u>	Deletes a string in a list box
<u>LB_DIR</u>	Adds file names to a list box
<u>LB_FINDSTRING</u>	Finds a string in a list box
<u>LB_GETCARETINDEX</u>	Gets index of list-box item with focus rectangle
<u>LB_GETCOUNT</u>	Gets the number of items in a list box
<u>LB_GETCURSEL</u>	Gets index of selected item in a list box
<u>LB_GETHORIZONTALEXTENT</u>	Gets the horizontal extent of a list box
<u>LB_GETITEMDATA</u>	Retrieves value associated with a list-box item
<u>LB_GETITEMHEIGHT</u>	Retrieves the height of items in a list box
<u>LB_GETITEMRECT</u>	Retrieves the bounding rectangle for an item
<u>LB_GETSEL</u>	Retrieves the selection state of an item
<u>LB_GETSELCOUNT</u>	Retrieves a count of selected list-box items
<u>LB_GETSELITEMS</u>	Lists item numbers of selected list-box items
<u>LB_GETTEXT</u>	Gets a string from a list box
<u>LB_GETTEXTLEN</u>	Gets the length of a string in a list box
<u>LB_GETTOPINDEX</u>	Retrieves index of first visible list-box item
<u>LB_INSERTSTRING</u>	Inserts a string into a list box
<u>LB_RESETCONTENT</u>	Removes all items from a list box
<u>LB_SELECTSTRING</u>	Selects a string in a list box
<u>LB_SELITEMRANGE</u>	Selects consecutive items in a list box
<u>LB_SETCARETINDEX</u>	Sets the focus rectangle in a list box
<u>LB_SETCOLUMNWIDTH</u>	Sets the width of columns in a list box
<u>LB_SETCURSEL</u>	Selects a string in a list box
<u>LB_SETHORIZONTALEXTENT</u>	Sets the horizontal extent of a list box
<u>LB_SETITEMDATA</u>	Associates a value with a list-box item
<u>LB_SETITEMHEIGHT</u>	Sets the height of items in a list box
<u>LB_SETSEL</u>	Selects a string in a multi-selection list box
<u>LB_SETTABSTOPS</u>	Sets tab stops in a list box
<u>LB_SETTOPINDEX</u>	Ensures that a list-box item is visible

New 3.1 messages

<u>CB_FINDSTRINGEXACT</u>	Finds a string in the list box of a combo box
<u>CB_GETDROPPEDCONTROLRECT</u>	Gets rectangle of combo-box drop-down list box
<u>CB_GETDROPPEDSTATE</u>	Determines if a combo box's list box is visible
<u>CB_GETEXTENDEDUI</u>	Selects the default or extended user interface
<u>CB_GETITEMHEIGHT</u>	Retrieves the height of items in a combo box
<u>CB_SETEXTENDEDUI</u>	Sets the default or extended user interface
<u>CB_SETITEMHEIGHT</u>	Sets the height of items in a combo box
<u>EM_GETFIRSTVISIBLELINE</u>	Gets index of top line in an edit control
<u>EM_GETPASSWORDCHAR</u>	Retrieves edit-control password character
<u>EM_GETWORDBREAKPROC</u>	Retrieves edit-control wordwrap function
<u>EM_SETREADONLY</u>	Sets the read-only state of an edit control
<u>EM_SETWORDBREAKPROC</u>	Provides custom word breaks in edit controls
<u>LB_FINDSTRINGEXACT</u>	Finds a string in a list box
<u>LB_GETCARETINDEX</u>	Gets index of list-box item with focus rectangle
<u>LB_GETITEMHEIGHT</u>	Retrieves the height of items in a list box
<u>LB_SETCARETINDEX</u>	Sets the focus rectangle in a list box
<u>LB_SETITEMHEIGHT</u>	Sets the height of items in a list box
<u>STM_GETICON</u>	Gets icon handle associated with an icon control
<u>STM_SETICON</u>	Associates an icon handle with an icon control
<u>WM_CHOOSEFONT_GETLOGFONT</u>	Retrieves LOGFONT for Font Dialog
<u>WM_COMMNOTIFY</u>	Notifies a window about the status of its queues
<u>WM_DROPFILES</u>	Indicates that a file has been dropped
<u>WM_PALETTEISCHANGING</u>	Indicates that the palette is changing
<u>WM_POWER</u>	Indicates the system is entering suspended mode
<u>WM_QUEUESYNC</u>	Delimits CBT messages
<u>WM_SYSTEMERROR</u>	Indicates a system error has occurred
<u>WM_WINDOWPOSCHANGED</u>	Notifies window of size or position change
<u>WM_WINDOWPOSCHANGING</u>	Notifies window of new size or position

## BM\_GETCHECK (2.x)

```
BM_GETCHECK
wParam = 0; /* not used, must be zero */
lParam = 0L; /* not used, must be zero */
```

An application sends a BM\_GETCHECK message to retrieve the check state of a button.

### Parameters

This message has no parameters.

### Returns

The return value from a button created with the BS\_AUTOCHECKBOX, BS\_AUTORADIOBUTTON, BS\_AUTO3STATE, BS\_CHECKBOX, BS\_RADIOBUTTON, or BS\_3STATE style may be one of the following values:

Value	Meaning
0	Button state is unchecked.
1	Button state is checked.
2	Button state is indeterminate (applies only if the button has the BS_3STATE or BS_AUTO3STATE style).

If the button has any other style, the return value is 0.

### Example

This example determines if the ID\_MYCHECKBOX control is currently checked:

```
int checked;

checked = (int) SendDlgItemMessage(hwndDlg, ID_MYCHECKBOX,
    BM_GETCHECK, 0, 0);
```

### See Also

BM\_GETSTATE, BM\_SETCHECK, SendDlgItemMessage

## BM\_GETSTATE (2.x)

BM\_GETSTATE

```
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends a BM\_GETSTATE message to retrieve the state of a button.

Parameters

This message has no parameters.

Returns

The return value specifies the current state of the button. You can use the following masks to extract information about the state:

Mask	Description
0x0003	Specifies the check state (radio buttons and check boxes only). A value of 0 indicates the button is unchecked. A value of 1 indicates the button is checked. A radio button is checked when it contains a dot; a check box is checked when it contains an X. A value of 2 indicates the check state is indeterminate (3-state check boxes only). The state of a 3-state check box is indeterminate when it is grayed.
0x0004	Specifies the highlight state. A nonzero value indicates that the button is highlighted. A button is highlighted when the user presses and holds the left mouse button. The highlighting is removed when the user releases the mouse button.
0x0008	Specifies the focus state. A nonzero value indicates that the button has the focus.

Example

This example determines whether a button currently has the focus:

```
#define BFFOCUS 0x0008
```

```
DWORD dwResult;
```

```
dwResult = SendMessage(hdlg, ID_MYBUTTON, BM_GETSTATE, 0, 0);  
if (dwResult & BFFOCUS)
```

```
    /* button has the focus */
```

See Also

BM\_GETCHECK, BM\_SETSTATE

## BM\_SETCHECK (2.x)

BM\_SETCHECK

```
wParam = (WPARAM) fCheck; /* check state */  
lParam = 0L; /* not used, must be zero */
```

An application sends a BM\_SETCHECK message to set the check state of a button.

Parameter	Description								
fCheck	Value of wParam. Specifies the check state. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Set the button state to unchecked.</td></tr><tr><td>1</td><td>Set the button state to checked.</td></tr><tr><td>2</td><td>Set the button state to indeterminate. This value can be used only if the button has the BS_3STATE or BS_AUTO3STATE style.</td></tr></tbody></table>	Value	Meaning	0	Set the button state to unchecked.	1	Set the button state to checked.	2	Set the button state to indeterminate. This value can be used only if the button has the BS_3STATE or BS_AUTO3STATE style.
Value	Meaning								
0	Set the button state to unchecked.								
1	Set the button state to checked.								
2	Set the button state to indeterminate. This value can be used only if the button has the BS_3STATE or BS_AUTO3STATE style.								

### Returns

The return value is always zero.

### Comments

The BM\_SETCHECK message has no effect on push buttons.

### Example

This example places a dot inside a radio button:

```
SendMessage(hDlg, ID_MYRADIOBUTTON, BM_SETCHECK, TRUE, 0);
```

### See Also

BM\_GETCHECK, BM\_SETSTATE

## BM\_SETSTATE (2.x)

BM\_SETSTATE

```
wParam = (WPARAM) fState; /* highlight state */  
lParam = 0L; /* not used, must be zero */
```

An application sends a BM\_SETSTATE message to set the highlight state of a button.

Parameter	Description
fState	Value of wParam. Specifies whether the button is to be highlighted. A nonzero value highlights the button. A zero value removes any highlighting.

Returns

The return value is always zero.

Comments

Highlighting affects the exterior of a button. It has no effect on the check state of a radio button or check box.

A button is automatically highlighted when the user presses and holds the left mouse button. The highlighting is removed when the user releases the mouse button.

Example

This example highlights and then removes highlighting from a push button, simulating the visual effect of a user clicking the button:

```
SendDlgItemMessage(hdlg, ID_MYPUSHBUTTON, BM_SETSTATE, TRUE, 0);
```

```
/*  
 * Perform some action; then remove the highlighting,  
 * thereby returning it to its normal state.  
 */
```

```
SendDlgItemMessage(hdlg, ID_MYPUSHBUTTON, BM_SETSTATE, FALSE, 0);
```

See Also

BM\_GETSTATE, BM\_SETCHECK



## BM\_SETSTYLE (2.x)

### BM\_SETSTYLE

```
wParam = (WPARAM) LOWORD(dwStyle); /* style */  
lParam = MAKELPARAM(fRedraw, 0); /* redraw flag */
```

An application sends a BM\_SETSTYLE message to change the style of a button.

Parameter	Description
dwStyle	Value of wParam. Specifies the button style. For a list of possible buttons styles, see the <a href="#">Button styles</a> topic.
fRedraw	Value of the low-order word of lParam. Specifies whether the button is to be redrawn. A value of TRUE redraws the button. A value of FALSE does not redraw the button.

### Returns

The return value is always zero.

### Comments

Unlike [BM\\_SETCHECK](#) and [BM\\_SETSTATE](#), [BM\\_SETSTYLE](#) does not have a corresponding message to retrieve the current style. Use the [GetWindowLong](#) function with the [GWL\\_STYLE](#) offset to retrieve the complete button style. The low word of the complete button style is the button-specific style.

An application should not attempt to change a button's type (for example, changing a radio button to a check box).

### Example

This example sends a BM\_SETSTYLE message to make a button become the default push button:

```
SendDlgItemMessage(hDlg, ID_MY PUSHBUTTON, BM_SETSTYLE,  
    (WPARAM) BS_DEFPUSHBUTTON, TRUE);
```

### See Also

[GetWindowLong](#)

## CB\_ADDSTRING (3.0)

```
CB_ADDSTRING  
wParam = 0; /* not used, must be zero */  
lParam = (LPARAM) (LPCSTR) lpsz; /* address of string to add */
```

An application sends a `CB_ADDSTRING` message to add a string to the list box of a combo box. If the combo box does not have the `CBS_SORT` style, the string is added to the end of the list. Otherwise, the string is inserted into the list and the list is sorted.

Parameter	Description
<code>lpsz</code>	Value of <code>lParam</code> . Points to the null-terminated string to be added. If the combo box was created with an owner-drawn style but without the <code>CBS_HASSTRINGS</code> style, the value of the <code>lpsz</code> parameter is stored rather than the string it would otherwise point to.

### Returns

The return value is the zero-based index to the string in the list box of the combo box. The return value is `CB_ERR` if an error occurs; the return value is `CB_ERRSPACE` if insufficient space is available to store the new string.

### Comments

If an owner-drawn combo box was created with the `CBS_SORT` style but not the `CBS_HASSTRINGS` style, the `WM_COMPAREITEM` message is sent one or more times to the owner of the combo box so that the new item can be properly placed in the list.

To insert a string into a specific location within the list, use the `CB_INSERTSTRING` message.

### Example

This example adds the string "my string" to the list box of a combo box:

```
DWORD dwIndex;
```

```
dwIndex = SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
    CB_ADDSTRING, 0, (LPARAM) ((LPSTR) "my string"));
```

### See Also

CB\_INSERTSTRING, WM\_COMPAREITEM, CB\_DIR

## CB\_DELETESTRING (3.0)

```
CB_DELETESTRING  
wParam = (WPARAM) index; /* item to delete */  
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_DELETESTRING message to delete a string in the list box of a combo box.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the string to delete.

### Returns

The return value is a count of the strings remaining in the list. The return value is CB\_ERR if the index parameter specifies an index greater than the number of items in the list.

### Comments

If the combo box was created with an owner-drawn style but without the CBS\_HASSTRINGS style, a WM\_DELETEITEM message is sent to the owner of the combo box so that the application can free any additional data associated with the item.

### Example

This example deletes the first string in a combo box:

```
DWORD dwRemaining;
```

```
dwRemaining = SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
    CB_DELETESTRING, 0, 0);
```

### See Also

WM\_DELETEITEM, CB\_RESETCONTENT

## CB\_DIR (3.0)

CB\_DIR

```
wParam = (WPARAM) (UINT) uAttrs; /* file attributes*/  
lParam = (LPARAM) (LPCSTR) lpszFileSpec; /* address of filename */
```

An application sends a CB\_DIR message to add a list of filenames to the list box of a combo box.

Parameter	Description																		
uAttrs	Value of wParam. Specifies the attributes of the files to be added to the list box. It can be any combination of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>DDL_READWRITE</td><td>File can be read from or written to.</td></tr><tr><td>DDL_READONLY</td><td>File can be read from but not written to.</td></tr><tr><td>DDL_HIDDEN</td><td>File is hidden and does not appear in a directory listing.</td></tr><tr><td>DDL_SYSTEM</td><td>File is a system file.</td></tr><tr><td>DDL_DIRECTORY</td><td>The name pointed to by the lpszFileSpec parameter specifies a directory.</td></tr><tr><td>DDL_ARCHIVE</td><td>File has been archived.</td></tr><tr><td>DDL_DRIVES</td><td>All drives that match the name specified by the lpszFileSpec parameter are included. If the DDL_DRIVES flag is set, the DDL_EXCLUSIVE flag is set automatically. Therefore, to create a directory listing that includes drives and files, the developer must send this message twice: once with the DDL_DRIVES flag set and once with the flags for the rest of the list.</td></tr><tr><td>DDL_EXCLUSIVE</td><td>Exclusive flag. If the exclusive flag is set, only files of the specified type are listed. Otherwise, files of the specified type are listed in addition to files that do not match the specified type.</td></tr></tbody></table>	Value	Meaning	DDL_READWRITE	File can be read from or written to.	DDL_READONLY	File can be read from but not written to.	DDL_HIDDEN	File is hidden and does not appear in a directory listing.	DDL_SYSTEM	File is a system file.	DDL_DIRECTORY	The name pointed to by the lpszFileSpec parameter specifies a directory.	DDL_ARCHIVE	File has been archived.	DDL_DRIVES	All drives that match the name specified by the lpszFileSpec parameter are included. If the DDL_DRIVES flag is set, the DDL_EXCLUSIVE flag is set automatically. Therefore, to create a directory listing that includes drives and files, the developer must send this message twice: once with the DDL_DRIVES flag set and once with the flags for the rest of the list.	DDL_EXCLUSIVE	Exclusive flag. If the exclusive flag is set, only files of the specified type are listed. Otherwise, files of the specified type are listed in addition to files that do not match the specified type.
Value	Meaning																		
DDL_READWRITE	File can be read from or written to.																		
DDL_READONLY	File can be read from but not written to.																		
DDL_HIDDEN	File is hidden and does not appear in a directory listing.																		
DDL_SYSTEM	File is a system file.																		
DDL_DIRECTORY	The name pointed to by the lpszFileSpec parameter specifies a directory.																		
DDL_ARCHIVE	File has been archived.																		
DDL_DRIVES	All drives that match the name specified by the lpszFileSpec parameter are included. If the DDL_DRIVES flag is set, the DDL_EXCLUSIVE flag is set automatically. Therefore, to create a directory listing that includes drives and files, the developer must send this message twice: once with the DDL_DRIVES flag set and once with the flags for the rest of the list.																		
DDL_EXCLUSIVE	Exclusive flag. If the exclusive flag is set, only files of the specified type are listed. Otherwise, files of the specified type are listed in addition to files that do not match the specified type.																		
lpszFileSpec	Value of lParam. Points to the null-terminated string that specifies the filename to add to the list. If the filename contains any wildcards (for example, *.*), all files that match and have the attributes specified by the uAttrs parameter will be added to the list.																		

### Returns

The return value is the zero-based index of the last filename added to the list. The return value is CB\_ERR if an error occurs. The return value is CB\_ERRSPACE if insufficient space is available to store the new strings.

### Example

This example adds the names of all available drives to a combo box:

```
DWORD dwIndexLastItem;
```

```
dwIndexLastItem = SendDlgItemMessage(hDlg, ID_MYCOMBOBOX, CB_DIR,  
0x4000 | 0x8000, (LPARAM) ((LPSTR) ""));
```

### See Also

DlgDirList, DlgDirListComboBox, CB\_ADDSTRING, CB\_INSERTSTRING

## CB\_FINDSTRING (3.0)

CB\_FINDSTRING

```
wParam = (WPARAM) indexStart; /* item before start of search */  
lParam = (LPARAM) (LPCSTR) lpszFind; /* address of prefix string */
```

An application sends a CB\_FINDSTRING message to search the list box of a combo box for an item that begins with the characters in a specified string.

Parameter	Description
indexStart	Value of wParam. Specifies the zero-based index of the item before the first item to be searched. When the search reaches the bottom of the list box, it continues from the top of the list box back to the item specified by the indexStart parameter. If indexStart is -1, the entire list box is searched from the beginning.
lpszFind	Value of lParam. Points to the null-terminated string that contains the prefix to search for. The search is not case-sensitive, so this string can contain any combination of uppercase and lowercase letters.

### Returns

The return value is the zero-based index of the matching item, or it is CB\_ERR if the search was unsuccessful.

### Comments

If the combo box was created with an owner-drawn style but without the CBS\_HASSTRINGS style, the action taken by the CB\_FINDSTRING message depends on whether the CBS\_SORT style is used. If the CBS\_SORT style is used, WM\_COMPAREITEM messages are sent to the owner of the combo box to determine which item matches the specified string. Otherwise, CB\_FINDSTRING attempts to match the doubleword value against the value of the lpszFind parameter.

### Example

This example searches for the string "my string" in a combo box and copies it, if found, to the szBuf buffer:

```
char szBuf[20];  
DWORD dwIndex;  
  
dwIndex = SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
    CB_FINDSTRING, 0, (LPARAM) ((LPSTR) "my string"));  
if (dwIndex != CB_ERR)  
    SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
        CB_GETLBTEXT, (WPARAM) dwIndex, (LPARAM) ((LPSTR) szBuf));
```

### See Also

CB\_FINDSTRINGEXACT, CB\_SELECTSTRING, WM\_COMPAREITEM

## CB\_FINDSTRINGEXACT (3.1)

### CB\_FINDSTRINGEXACT

```
wParam = (WPARAM) indexStart; /* item before start of search */  
lParam = (LPARAM) (LPCSTR) lpszFind; /* address of prefix string */
```

An application sends a CB\_FINDSTRINGEXACT message to find the first list box string (in a combo box) that matches the string specified in the lpszFind parameter.

Parameter	Description
indexStart	Value of wParam. Specifies the zero-based index of the item before the first item to be searched. When the search reaches the bottom of the list box, it continues from the top of the list box back to the item specified by the indexStart parameter. If indexStart is -1, the entire list box is searched from the beginning.
lpszFind	Value of lParam. Points to the null-terminated string to search for. This string can contain a complete filename, including the extension. The search is not case-sensitive, so this string can contain any combination of uppercase and lowercase letters.

### Returns

The return value is the zero-based index of the matching item, or it is CB\_ERR if the search was unsuccessful.

### Comments

If the combo box was created with an owner-drawn style but without the CBS\_HASSTRINGS style, the action taken by the CB\_FINDSTRINGEXACT message depends on whether the CBS\_SORT style is used. If the CBS\_SORT style is used, WM\_COMPAREITEM messages are sent to the owner of the combo box to determine which item matches the specified string. Otherwise, CB\_FINDSTRINGEXACT attempts to match the doubleword value against the value of the lpszFind parameter.

### Example

This example searches for the string "my string" in a combo box and copies it, if found, to the szBuf buffer:

```
char szBuf[20];  
DWORD dwIndex;  
  
dwIndex = SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
    CB_FINDSTRINGEXACT, 0, (LPARAM) ((LPSTR) "my string"));  
if (dwIndex != CB_ERR)  
    SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
        CB_GETLBTEXT, (WPARAM) dwIndex, (LPARAM) ((LPSTR) szBuf));
```

### See Also

CB\_FINDSTRING, CB\_SELECTSTRING, WM\_COMPAREITEM

## CB\_GETCOUNT (3.0)

```
CB_GETCOUNT  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_GETCOUNT message to retrieve the number of items in the list box of a combo box.

### Parameters

This message has no parameters.

### Returns

The return value is the number of items in the list box.

### Comments

The returned count is one greater than the index value of the last item (the index is zero-based).

### Example

This example retrieves the number of items in a combo box:

```
WORD cListItems;
```

```
cListItems = (WORD) SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
    CB_GETCOUNT, 0, 0);
```

## CB\_GETCURSEL (3.0)

```
CB_GETCURSEL  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_GETCURSEL message to retrieve the index of the currently selected item, if any, in the list box of a combo box.

### Parameters

This message has no parameters.

### Returns

The return value is the zero-based index of the currently selected item, or it is CB\_ERR if no item is selected.

### Example

This example retrieves the index of the currently selected string in the list box of a combo box and then retrieves that string:

```
char szBuf[20];  
DWORD dwIndex;  
  
dwIndex = SendDlgItemMessage(hdlg, ID_MYCOMBOBOX, CB_GETCURSEL, 0, 0);  
if (dwIndex != CB_ERR)  
    SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
        CB_GETLBTEXT, (WPARAM) dwIndex, (LPARAM) ((LPSTR) szBuf));
```

### See Also

CB\_SELECTSTRING, CB\_SETCURSEL



## CB\_GETDROPPEDCONTROLRECT (3.1)

CB\_GETDROPPEDCONTROLRECT

```
wParam = 0; /* not used, must be zero */  
lParam = (LPARAM) (RECT FAR*) lprc; /* address of RECT structure */
```

An application sends a CB\_GETDROPPEDCONTROLRECT message to retrieve the screen coordinates of the visible (dropped-down) list box of a combo box.

Parameter	Description
lprc	Value of lParam. Points to the <u>RECT</u> structure that is to receive the coordinates.

Returns

The return value is always CB\_OKAY.

Example

This example retrieves the bounding rectangle of the list box of a combo box:

```
RECT rcl;
```

```
SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
CB_GETDROPPEDCONTROLRECT, 0, (DWORD) ((LPRECT) &rcl));
```

## CB\_GETDROPPEDSTATE (3.1)

```
CB_GETDROPPEDSTATE  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_GETDROPPEDSTATE message to determine whether the list box of a combo box is visible (dropped down).

### Parameters

This message has no parameters.

### Returns

The return value is nonzero if the list box is visible; otherwise, it is zero.

### Example

This example determines whether the list box of a combo box is visible:

```
BOOL fDropped;
```

```
fDropped = (BOOL) SendMessage(hdlg, ID_MYCOMBOBOX,  
    CB_GETDROPPEDSTATE, 0, 0L);
```

### See Also

CB\_SHOWDROPDOWN

## CB\_GETEDITSEL (2.x)

```
CB_GETEDITSEL  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_GETEDITSEL message to retrieve the starting and ending character positions of the current selection in the edit control of a combo box.

### Parameters

This message has no parameters.

### Returns

The return value is a doubleword value that contains the starting position in the low-order word and the position of the first nonselected character after the end of the selection in the high-order word.

### Example

This example retrieves the selection positions of the edit control of a combo box, and converts them into starting and ending positions:

```
DWORD dwResult;  
WORD wStart, wEnd;  
  
dwResult = SendMessage(hdlg, ID_MYCOMBOBOX,  
    CB_GETEDITSEL, 0, 0);  
wStart = LOWORD(dwResult);  
wEnd   = HWORD(dwResult);
```

### See Also

CB\_SETEDITSEL

## CB\_GETEXTENDEDUI (3.1)

```
CB_GETEXTENDEDUI  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_GETEXTENDEDUI message to determine whether a combo box has the default user interface or the extended user interface.

### Parameters

This message has no parameters.

### Returns

The return value is nonzero if the combo box has the extended user interface; otherwise, it is zero.

### Comments

The extended user interface differs from the default user interface in the following ways:

- Clicking the static control displays the list box (CBS\_DROPDOWNLIST style only).
- Pressing the DOWN ARROW key displays the list box (F4 is disabled).
- Scrolling in the static control is disabled when the item list is not visible (arrow keys are disabled).

### Example

This example determines whether a combo box has the extended user interface:

```
BOOL fExtended;
```

```
fExtended = (BOOL) SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
    CB_GETEXTENDEDUI, 0, 0L);
```

### See Also

CB\_SETEXTENDEDUI

## CB\_GETITEMDATA (3.0)

```
CB_GETITEMDATA  
wParam = (WPARAM) index; /* item index */  
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_GETITEMDATA message to a combo box to retrieve the application-supplied doubleword value associated with the specified item in the combo box. (This is the value in the lParam parameter of a CB\_SETITEMDATA message.)

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the item.

### Returns

The return value is the doubleword value associated with the item, or it is CB\_ERR if an error occurs.

### See Also

CB\_SETITEMDATA

## CB\_GETITEMHEIGHT (3.1)

```
CB_GETITEMHEIGHT
wParam = (WPARAM) index; /* item index */
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_GETITEMHEIGHT message to retrieve the height of list items in a combo box.

Parameter	Description
index	Value of wParam. Specifies the component of the combo box whose height is to be retrieved. If the index parameter is -1, the height of the edit-control (or static-text) portion of the combo box is retrieved. If the combo box has the CBS_OWNERDRAWVARIABLE style, index specifies the zero-based index of the list item whose height is to be retrieved. Otherwise, index should be set to zero.

### Returns

The return value is the height, in pixels, of the list items in a combo box. The return value is the height of the item specified by the index parameter if the combo box has the CBS\_OWNERDRAWVARIABLE style. The return value is the height of the edit-control (or static-text) portion of the combo box if index is -1. The return value is CB\_ERR if an error occurred.

### Example

This example sends a CB\_GETITEMHEIGHT message to retrieve the height of the list items in a combo box:

```
LRESULT lrHeight;
```

```
lrHeight = SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
    CB_GETITEMHEIGHT, 0, 0L);
```

### See Also

CB\_SETITEMHEIGHT, WM\_MEASUREITEM

## CB\_GETLBTEXT (3.0)

CB\_GETLBTEXT

```
wParam = (WPARAM) index; /* item index */  
lParam = (LPARAM) (LPCSTR) lpszBuffer; /* address of buffer */
```

An application sends a CB\_GETLBTEXT message to retrieve a string from the list box of a combo box.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the string to retrieve.
lpszBuffer	Value of lParam. Points to the buffer that receives the string. The buffer must have sufficient space for the string and a terminating null character. A <u>CB_GETLBTEXTLEN</u> message can be sent before the CB_GETLBTEXT message to retrieve the length, in bytes, of the string.

### Returns

The return value is the length of the string, in bytes, excluding the terminating null character. If the index parameter does not specify a valid index, the return value is CB\_ERR.

### Comments

If the combo box was created with an owner-drawn style but without the CBS\_HASSTRINGS style, the buffer pointed to by the lpszBuffer parameter of the message receives the doubleword value associated with the item.

### Example

This example retrieves the length of the first item in the list box of a combo box, allocates sufficient memory for the string, and sends a CB\_GETLBTEXT message to retrieve the string:

```
DWORD cbItemString;  
PSTR psz;  
  
cbItemString = SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
    CB_GETLBTEXTLEN, 0, 0);  
if (cbItemString != CB_ERR) {  
    psz = (PSTR) LocalAlloc(LMEM_FIXED, (WORD) cbItemString);  
    SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
        CB_GETLBTEXT, 0, (LPARAM) ((LPSTR) psz));  
}
```

### See Also

CB\_GETLBTEXTLEN

## CB\_GETLBTEXTLEN (3.0)

```
CB_GETLBTEXTLEN  
wParam = (WPARAM) index; /* item index */  
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_GETLBTEXTLEN message to retrieve the length of a string in the list box of a combo box.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the string.

### Returns

The return value is the length of the string, in bytes, excluding the terminating null character. If the index parameter does not specify a valid index, the return value is CB\_ERR.

### Example

This example retrieves the length of the first item in the list box of a combo box:

```
DWORD cbItemString;
```

```
cbItemString = SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
    CB_GETLBTEXTLEN, 0, 0);
```

See Also

CB\_GETLBTEXT



## CB\_INSERTSTRING (3.0)

CB\_INSERTSTRING

```
wParam = (WPARAM) index; /* item index */  
lParam = (LPARAM) (LPCSTR) lpsz; /* address of string to insert */
```

An application sends a CB\_INSERTSTRING message to insert a string into the list box of a combo box. Unlike the CB\_ADDSTRING message, the CB\_INSERTSTRING message does not cause a list with the CBS\_SORT style to be sorted.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the position at which to insert the string. If this parameter is -1, the string is added to the end of the list.
lpsz	Value of lParam. Points to the null-terminated string that is to be inserted. If the combo box was created with an owner-drawn style but without the CBS_HASSTRINGS style, the value of the lpsz parameter is stored rather than the string it would otherwise point to.

### Returns

The return value is the index of the position at which the string was inserted. The return value is CB\_ERR if an error occurs. The return value is CB\_ERRSPACE if insufficient space is available to store the new string.

### Example

This example inserts the string "my string" into the third position in the list box of a combo box:

```
SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,  
CB_INSERTSTRING, 2, (LPARAM) ((LPSTR) "my string"));
```

See Also

CB\_ADDSTRING, CB\_DIR

## CB\_LIMITTEXT (3.0)

### CB\_LIMITTEXT

```
wParam = (WPARAM) cchLimit; /* maximum number of characters */  
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_LIMITTEXT message to limit the length of the text that the user may type in the edit control of a combo box.

Parameter	Description
cchLimit	Value of wParam. Specifies the length, in bytes, of the text the user can enter. If this parameter is zero, the text length is set to 65,535 bytes.

### Returns

The return value is 1 if the message is successful. If this message is sent to a combo box with the style CBS\_DROPDOWNLIST, the return value is CB\_ERR.

### Comments

If the combo box does not have the style CBS\_AUTOHSCROLL, setting the text limit to be larger than the size of the edit control has no effect.

The CB\_LIMITTEXT message limits only the text the user can enter. It has no effect on any text already in the edit control when the message is sent, nor does it affect the length of the text copied to the edit control when a string in the list box is selected.

### Example

This example limits the text of the edit control of a combo box to five characters:

```
SendDlgItemMessage(hDlg, ID_MYCOMBOBOX, CB_LIMITTEXT, 5, 0);
```

## CB\_RESETCONTENT (3.0)

```
CB_RESETCONTENT  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_RESETCONTENT message to remove all items from the list box and edit control of a combo box.

### Parameters

This message has no parameters.

### Returns

The return value is always CB\_OKAY.

### Comments

If the combo box was created with an owner-drawn style but without the CBS\_HASSTRINGS style, the owner of the combo box receives a WM\_DELETEITEM message for each item in the combo box.

### Example

This example removes all items from the list box and edit control of a combo box:

```
SendDlgItemMessage(hdlg, ID_MYCOMBOBOX, CB_RESETCONTENT, 0, 0);
```

### See Also

CB\_DELETESTRING, WM\_DELETEITEM

## CB\_SELECTSTRING (3.0)

```
CB_SELECTSTRING
wParam = (WPARAM) indexStart; /* item before first selection */
lParam = (LPARAM) (LPCSTR) lpszSelect; /* address of prefix string  *
```

An application sends a CB\_SELECTSTRING message to search the list box of a combo box for an item that begins with the characters in a specified string. If a matching item is found, the item is selected and copied to the edit control.

Parameter	Description
indexStart	Value of wParam. Specifies the zero-based index of the item before the first item to be searched. When the search reaches the bottom of the list box, it continues from the top of the list box back to the item specified by the indexStart parameter. If indexStart is -1, the entire list box is searched from the beginning.
lpszSelect	Value of lParam. Points to the null-terminated string that contains the prefix to search for. The search is not case-sensitive, so this string can contain any combination of uppercase and lowercase letters.

### Returns

The return value is the index of the selected item if the string was found. The return value is CB\_ERR and the current selection is not changed if the search was unsuccessful.

### Comments

A string is selected only if its initial characters (from the starting point) match the characters in the prefix string.

If the combo box was created with an owner-drawn style but without the CBS\_HASSTRINGS style, the action taken by the CB\_SELECTSTRING message depends on whether the CBS\_SORT style is used. If the CBS\_SORT style is used, WM\_COMPAREITEM messages are sent to the owner of the combo box to determine which item matches the specified string. Otherwise, CB\_SELECTSTRING attempts to match the doubleword value against the value of the lpszFind parameter.

### Example

This example searches the entire list box of a combo box for the string "my string" and, if the string is found, selects it:

```
DWORD dwIndexFoundString;

dwIndexFoundString = SendDlgItemMessage(hdlg, ID_MYCOMBOBOX,
    CB_SELECTSTRING, -1, (LPARAM) ((LPSTR) "my string"));
```

### See Also

CB\_FINDSTRING, CB\_FINDSTRINGEXACT, CB\_SETCURSEL, WM\_COMPAREITEM

## CB\_SETCURSEL (3.0)

```
CB_SETCURSEL  
wParam = (WPARAM) index; /* item index */  
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_SETCURSEL message to select a string in the list box of a combo box. If necessary, the list box scrolls the string into view (if the list box is visible). The text in the edit control of the combo box is changed to reflect the new selection. Any previous selection in the list box is removed.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the string to select. If the index parameter is -1, any current selection in the list box is removed and the edit control is cleared.

### Returns

The return value is the index of the item selected if the message is successful. The return value is CB\_ERR if the index parameter is greater than the number of items in the list or if index is set to -1 (which clears the selection).

### Example

This example retrieves the number of items in the list box of a combo box and sends a CB\_SETCURSEL message to select the last item in the list:

```
WORD cListItems;  
  
cListItems = (WPARAM) SendMessage(hdlg,  
    ID_MYCOMBOBOX, CB_GETCOUNT, 0, 0);  
SendMessage(hdlg, ID_MYCOMBOBOX,  
    CB_SETCURSEL,  
    cListItems - 1, /* zero-based index, so subtract one from total */  
    0);
```

### See Also

CB\_GETCURSEL, CB\_SELECTSTRING

## CB\_SETEDITSEL (3.0)

```
CB_SETEDITSEL  
wParam = 0; /* not used, must be zero */  
lParam = MAKELPARAM((ichStart), (ichEnd)); /* start and end positions */  
/
```

An application sends a CB\_SETEDITSEL message to select characters in the edit control of a combo box.

Parameter	Description
ichStart	Value of the low-order word of lParam. Specifies the starting position. If this parameter is set to -1, the selection, if any, is removed.
ichEnd	Value of the high-order word of lParam. Specifies the ending position. If this parameter is set to -1, all text from the starting position to the last character in the edit control is selected.

### Returns

The return value is nonzero if the message is successful. It is CB\_ERR if the message is sent to a combo box with the CBS\_DROPDOWNLIST style.

### Comments

The positions are zero-based. To select the first character of the edit control, you specify a starting position of zero. The ending position is for the character just after the last character to select. For example, to select the first four characters of the edit control, you would use a starting position of 0 and an ending position of 4.

### Example

This example selects the first four characters of the edit control of a combo box:

```
SendMessage(hdlg, ID_MYCOMBOBOX,  
            CB_SETEDITSEL, 0, MAKELONG(0, 4));
```

### See Also

CB\_GETEDITSEL

## CB\_SETEXTENDEDUI (3.1)

CB\_SETEXTENDEDUI

```
wParam = (WPARAM) (BOOL) fExtended; /* extended UI flag */  
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_SETEXTENDEDUI message to select either the default user interface or the extended user interface for a combo box that has the CBS\_DROPDOWN or CBS\_DROPDOWNLIST style.

Parameter	Description
fExtended	Value of wParam. Specifies whether the combo box should use the extended user interface or the default user interface. A value of TRUE selects the extended user interface; a value of FALSE selects the standard user interface.

### Returns

The return value is CB\_OKAY if the operation is successful, or it is CB\_ERR if an error occurred.

### Comments

The extended user interface differs from the default user interface in the following ways:

- Clicking the static control displays the list box (CBS\_DROPDOWNLIST style only).
- Pressing the DOWN ARROW key displays the list box (F4 is disabled).
- Scrolling in the static control is disabled when the item list is not visible (the arrow keys are disabled).

### Example

This example selects the extended user interface for a combo box:

```
SendMessage(hdlg, ID_MYCOMBOBOX, CB_SETEXTENDEDUI,  
TRUE, 0L);
```

### See Also

CB\_GETEXTENDEDUI

## CB\_SETITEMDATA (3.0)

CB\_SETITEMDATA

```
wParam = (WPARAM) index; /* item index */  
lParam = (LPARAM) (DWORD) dwData; /* item data */
```

An application sends a CB\_SETITEMDATA message to set the doubleword value associated with the specified item in a combo box.

Parameter	Description
index	Value of wParam. Specifies the zero-based index to the item.
dwData	Value of lParam. Specifies the new value to be associated with the item.

Returns

The return value is CB\_ERR if an error occurs.

See Also

[CB\\_GETITEMDATA](#)



## CB\_SETITEMHEIGHT (3.1)

CB\_SETITEMHEIGHT

```
wParam = (WPARAM) index; /* item index */  
lParam = (LPARAM) (int) height; /* item height */
```

An application sends a CB\_SETITEMHEIGHT message to set the height of list items in a combo box or the height of the edit-control (or static-text) portion of a combo box.

Parameter	Description
index	Value of wParam. Specifies whether the height of list items or the height of the edit-control (or static-text) portion of the combo box is set. If the combo box has the <u>CBS_OWNERDRAWVARIABLE</u> style, the index parameter specifies the zero-based index of the list item whose height is to be set; otherwise, index must be zero and the height of all list items will be set. If index is -1, the height of the edit-control or static-text portion of the combo box is to be set.
height	Value of the low-order word of lParam. Specifies the height, in pixels, of the combo box component identified by index.

### Returns

The return value is CB\_ERR if the index or height is invalid.

### Comments

The height of the edit-control (or static-text) portion of the combo box is set independently of the height of the list items. An application must ensure that the height of the edit-control (or static-text) portion isn't smaller than the height of a particular list box item.

### Example

This example sends a CB\_SETITEMHEIGHT message to set the height of list items in a combo box:

```
LPARAM lrHeight;
```

```
SendMessage(hdlg, ID_MYCOMBOBOX, CB_SETITEMHEIGHT,  
0, lrHeight);
```

### See Also

CB\_GETITEMHEIGHT, WM\_MEASUREITEM

## CB\_SHOWDROPDOWN (3.0)

CB\_SHOWDROPDOWN

```
wParam = (WPARAM) (BOOL) fShow; /* the show/hide flag*/  
lParam = 0L; /* not used, must be zero */
```

An application sends a CB\_SHOWDROPDOWN message to show or hide the list box of a combo box that has the CBS\_DROPDOWN or CBS\_DROPDOWNLIST style.

Parameter	Description
fShow	Value of wParam. Specifies whether the drop-down list box is to be shown or hidden. A value of TRUE shows the list box. A value of FALSE hides the list box.

### Returns

The return value is always nonzero.

### Comments

This message has no effect on a combo box created with the CBS\_SIMPLE style.

### Example

This example shows the list box of a combo box:

```
SendMessage(hDlg, ID_MYCOMBOBOX, CB_SHOWDROPDOWN, TRUE, 0);
```

See Also

CB\_GETDROPPEDSTATE

## DM\_GETDEFID

```
DM_GETDEFID
wParam = 0; /* not used, must be zero */
lParam = 0L; /* not used, must be zero */
```

An application sends a DM\_GETDEFID message to get the identifier of the default push button for a dialog box.

### Parameters

This message has no parameters.

### Returns

The return value is a doubleword value. If the default push button has an identifier value, the high-order word contains DC\_HASDEFID and the low-order word contains the identifier value. The return value is zero if the default push button does not have an identifier value.

### Example

This example gets the identifier of the default push button of a dialog box:

```
DWORD dwResult;
WORD idDefPushButton;

dwResult = SendMessage(hDlg, DM_GETDEFID, 0, 0);
if (HIWORD(dwResult) == DC_HASDEFID)
    idDefPushButton = LOWORD(dwResult);
```

### See Also

DM\_SETDEFID

## DM\_SETDEFID (2.x)

DM\_SETDEFID

```
wIDPushBtn = wParam; /* identifier of new default push button */
```

An application sends a DM\_SETDEFID message to change the identifier of the default push button for a dialog box.

Parameter	Description
wIDPushBtn	Value of wParam. Specifies the identifier of the push button that will become the default.

Returns

The return value is always nonzero.

See Also

[DM\\_GETDEFID](#)

## EM\_CANUNDO (2.x)

```
EM_CANUNDO
wParam = 0; /* not used, must be zero */
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_CANUNDO message to determine whether an edit-control operation can be undone.

### Parameters

This message has no parameters.

### Returns

The return value is nonzero if the last edit operation can be undone, or it is zero if the last edit operation cannot be undone.

### Example

This example sends an EM\_CANUNDO message to determine whether the last edit-control operation can be undone and, if so, sends an EM\_UNDO message to undo the last operation:

```
if (SendDlgItemMessage(hdlg, ID_MYEDITCONTROL, EM_CANUNDO, 0, 0))
    SendDlgItemMessage(hdlg, ID_MYEDITCONTROL, EM_UNDO, 0, 0);
```

### See Also

EM\_EMPTYUNDOBUFFER, EM\_UNDO

## EM\_EMPTYUNDOBUFFER (3.0)

```
EM_EMPTYUNDOBUFFER  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_EMPTYUNDOBUFFER message to reset (clear) the undo flag of an edit control. The undo flag is set whenever an operation within the edit control can be undone.

### Parameters

This message has no parameters.

### Returns

This message does not return a value.

### Comments

The undo flag is automatically cleared whenever the edit control receives a WM\_SETTEXT or EM\_SETHANDLE message.

### Example

This example resets the undo flag of an edit control:

```
SendMessage(hdlg, ID_MYEDITCONTROL, EM_EMPTYUNDOBUFFER, 0, 0);
```

### See Also

EM\_CANUNDO, EM\_UNDO

## EM\_FMTLINES (2.x)

```
EM_FMTLINES
wParam = (WPARAM) (BOOL) fAddEOL; /* line break flag */
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_FMTLINES message to set the inclusion of soft line break characters on or off within a multiline edit control. A soft line break consists of two carriage returns and a linefeed inserted at the end of a line that is broken because of wordwrapping.

This message is processed only by multiline edit controls.

Parameter	Description
fAddEOL	Value of wParam. Specifies whether soft line break characters are to be inserted. A value of TRUE inserts the characters; a value of FALSE removes them.

### Returns

The return value is identical to the fAddEOL parameter.

### Comments

This message affects only the buffer returned by the EM\_GETHANDLE message and the text returned by the WM\_GETTEXT message. It has no effect on the display of the text within the edit control.

A line that ends with a hard line break is not affected by the EM\_FMTLINES message. A hard line break consists of one carriage return and a linefeed.

### Example

This example sends an EM\_FMTLINES message to turn off soft line breaks, then allocates a buffer for the text, and then retrieves the text by sending a WM\_GETTEXT message:

```
WPARAM cbText;
HGLOBAL hmem;
LPSTR lpstr;

SendMessage(hdlg, ID_MYEDITCONTROL,
EM_FMTLINES, FALSE, 0);

cbText = (WPARAM) SendMessage(hdlg, ID_MYEDITCONTROL,
WM_GETTEXTLENGTH, 0, 0);
cbText++; /* make room for the terminating null character */
hmem = (HGLOBAL) GlobalAlloc(GMEM_MOVEABLE, (DWORD) cbText);
lpstr = GlobalLock(hmem);
SendMessage(hdlg, ID_MYEDITCONTROL,
WM_GETTEXT, cbText, (LPARAM) lpstr);
```

### See Also

EM\_GETWORDBREAKPROC, EM\_SETWORDBREAKPROC

## EM\_GETFIRSTVISIBLELINE (3.1)

```
EM_GETFIRSTVISIBLELINE  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_GETFIRSTVISIBLELINE message to determine the topmost visible line in an edit control.

### Parameters

This message has no parameters.

### Returns

The return value is the zero-based index of the topmost visible line. For single-line edit controls, the return value is zero.

### Example

This example gets the index of the topmost visible line in an edit control:

```
int FirstVis;  
  
FirstVis = (int) SendMessage(hdlg, IDD_EDIT,  
    EM_GETFIRSTVISIBLELINE, 0, 0L);
```



## EM\_GETHANDLE (2.x)

```
EM_GETHANDLE
wParam = 0; /* not used, must be zero */
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_GETHANDLE message to retrieve a handle to the memory currently allocated for a multiline edit control. The handle is a local memory handle and can be used by any of the functions that take a local memory handle as a parameter.

This message is processed only by multiline edit controls.

Parameters

This message has no parameters.

Returns

The return value is a local memory handle identifying the buffer that holds the contents of the edit control. If an error occurs, such as sending the message to a single-line edit control, the return value is zero.

Comments

An application can send this message to a multiline edit control in a dialog box only if it created the dialog box with the DS\_LOCALEEDIT style flag set. If the DS\_LOCALEEDIT style is not set, the return value is still nonzero, but the return value will not be meaningful.

Example

This example sends an EM\_GETHANDLE message to a multiline edit control and calls the LocalSize function to determine the current size of the edit control using the handle returned by the EM\_GETHANDLE message:

```
HANDLE hmemMle;
```

```
WORD cbMle;
```

```
hmemMle = (HLOCAL) SendDlgItemMessage(hdlg, ID_MYEDITCONTROL,  
    EM_GETHANDLE, 0, 0);  
cbMle = LocalSize(hmemMle);
```

See Also

EM\_SETHANDLE

## EM\_GETLINE (2.x)

### EM\_GETLINE

```
wParam = (WPARAM) line; /* line number to retrieve */  
lParam = (LPARAM) (LPSTR) lpch; /* address of buffer for line */
```

An application sends an EM\_GETLINE message to retrieve a line of text from an edit control.

Parameter	Description
line	Value of wParam. Specifies the line number of the line to retrieve from a multiline edit control. Line numbers are zero-based; a value of zero specifies the first line. This parameter is ignored by a single-line edit control.
lpch	Value of lParam. Points to the buffer that receives a copy of the line. The first word of the buffer specifies the maximum number of bytes that can be copied to the buffer.

### Returns

The return value is the number of bytes actually copied. The return value is zero if the line number specified by the line parameter is greater than the number of lines in the edit control.

### Comments

The copied line does not contain a terminating null character.

### Example

This example sets the maximum size of the buffer, sends an EM\_GETLINE message to get the first line of the multiline edit control, and adds a terminating null character to the end of the retrieved line:

```
unsigned char szBuf[128];  
WORD cbText;  
  
*(WORD *) szBuf = sizeof(szBuf) - 1; /* sets the buffer size */  
cbText = (WORD) SendMessage(hdlg, ID_MYEDITCONTROL,  
    EM_GETLINE,  
    0, /* line number */  
    (DWORD) (LPSTR) szBuf); /* buffer address */  
szBuf[cbText] = '\\0'; /* terminating null character */
```

### See Also

EM\_LINELENGTH, WM\_GETTEXT

## EM\_GETLINECOUNT (2.x)

```
EM_GETLINECOUNT
wParam = 0; /* not used, must be zero */
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_GETLINECOUNT message to retrieve the number of lines in a multiline edit control.

This message is processed only by multiline edit controls.

### Parameters

This message has no parameters.

### Returns

The return value is an integer containing the number of lines in the multiline edit control. If no text is in the edit control, the return value is 1.

### Example

This example sends an EM\_GETLINECOUNT message to retrieve the number of lines in a multiline edit control and then sends an EM\_LINESCROLL message to scroll the edit control so that the last line is displayed at the top of the edit control.

```
int cLines;

cLines = (int) SendMessage(hdlg, ID_MYEDITCONTROL,
    EM_GETLINECOUNT, 0, 0);
SendMessage(hdlg, ID_MYEDITCONTROL,
    EM_LINESCROLL, 0, MAKELONG(cLines - 1, 0));
```

## EM\_GETMODIFY (2.x)

```
EM_GETMODIFY
wParam = 0; /* not used, must be zero */
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_GETMODIFY message to determine whether the contents of an edit control have been modified.

### Parameters

This message has no parameters.

### Returns

The return value is nonzero if the edit-control contents have been modified, or it is zero if the contents have remained unchanged.

### Comments

Windows maintains an internal flag indicating whether the contents of the edit control have been changed. This flag is cleared when the edit control is first created; or an EM\_SETMODIFY message can be sent to clear the flag.

### Example

This example sends an EM\_GETMODIFY message to determine whether the edit control has been modified and, if it has, retrieves the current contents of the edit control and clears the modification flag by sending an EM\_SETMODIFY message:

```
char szBuf[128];

if (SendDlgItemMessage(hdlg, ID_MYEDITCONTROL,
    EM_GETMODIFY, 0, 0)) {
    SendDlgItemMessage(hdlg, ID_MYEDITCONTROL,
        WM_GETTEXT, sizeof(szBuf), (LPARAM) ((LPSTR) szBuf));
    SendDlgItemMessage(hdlg, ID_MYEDITCONTROL,
        EM_SETMODIFY, FALSE, 0);
}
```

### See Also

EM\_SETMODIFY

## EM\_GETPASSWORDCHAR (3.1)

### EM\_GETPASSWORDCHAR

```
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_GETPASSWORDCHAR message to retrieve the password character displayed in an edit control when the user enters text.

#### Parameters

This message has no parameters.

#### Returns

The return value specifies the character to be displayed in place of the character typed by the user. The return value is NULL if no password character exists.

#### Comments

If the edit control is created with the ES\_PASSWORD style, the default password character is set to an asterisk (\*).

#### See Also

EM\_SETPASSWORDCHAR

## EM\_GETRECT (2.x)

EM\_GETRECT

```
wParam = 0; /* not used, must be zero */  
lParam = (LPARAM) (RECT FAR*) lprc; /* address of RECT structure */
```

An application sends an EM\_GETRECT message to retrieve the formatting rectangle of an edit control. The formatting rectangle is the limiting rectangle of the text. The limiting rectangle is independent of the size of the edit-control window.

Parameter	Description
lprc	Value of lParam. Points to the <u>RECT</u> structure that receives the formatting rectangle.

Returns

The return value is not a meaningful value.

Comments

The formatting rectangle of a multiline edit control can be modified by the EM\_SETRECT and EM\_SETRECTNP messages.

Example

This example sends an EM\_GETRECT message to retrieve the formatting rectangle of an edit control:

```
RECT rcl;
```

```
SendMessage(hdlg, ID_MYEDITCONTROL,  
EM_GETRECT, 0, (DWORD) ((LPRECT) &rcl));
```

See Also

EM\_SETRECT, EM\_SETRECTNP, RECT

## EM\_GETSEL (2.x)

```
EM_GETSEL
wParam = 0; /* not used, must be zero */
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_GETSEL message to get the starting and ending character positions of the current selection in an edit control.

### Parameters

This message has no parameters.

### Returns

The return value is a doubleword value that contains the starting position in the low-order word and the position of the first nonselected character after the end of the selection in the high-order word.

### Example

This example gets the selection positions of an edit control and converts them into starting and ending positions:

```
DWORD dwResult;
WORD wStart, wEnd;

dwResult = SendDlgItemMessage(hdlg, ID_MYCOMBOBOX, EM_GETSEL, 0, 0);
wStart = LOWORD(dwResult);
wEnd = HIWORD(dwResult);
```

### See Also

EM\_REPLACESEL, EM\_SETSEL

## EM\_GETWORDBREAKPROC (3.1)

```
EM_GETWORDBREAKPROC  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends the EM\_GETWORDBREAKPROC message to an edit control to retrieve the current wordwrap function.

### Parameters

This message has no parameters.

### Returns

The return value specifies the procedure-instance address of the application-defined wordwrap function. The return value is NULL if no wordwrap function exists.

### Comments

A wordwrap function scans a text buffer (which contains text to be sent to the display), looking for the first word that does not fit on the current display line. The wordwrap function places this word at the beginning of the next line on the display. A wordwrap function defines at what point Windows should break a line of text for multiline edit controls, usually at a space character that separates two words.

### See Also

[EM\\_FMTLINES](#), [EM\\_SETWORDBREAKPROC](#), [MakeProcInstance](#), [WordBreakProc](#)



## EM\_LIMITTEXT (2.x)

### EM\_LIMITTEXT

```
wParam = (WPARAM) cchMax; /* text length */  
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_LIMITTEXT message to limit the length of the text the user can enter into an edit control.

Parameter	Description
cchMax	Value of wParam. Specifies the length, in bytes, of the text the user can enter. If this parameter is zero, the text length is set to 65,535 bytes.

### Returns

This message does not return a value.

### Comments

The EM\_LIMITTEXT message limits only the text the user can enter. It has no effect on any text already in the edit control when the message is sent, nor does it affect the length of text copied to the edit control by the WM\_SETTEXT message.

If an application uses the WM\_SETTEXT message to place more text into an edit control than is specified in the EM\_LIMITTEXT message, the user can edit the entire contents of the edit control.

## EM\_LINEFROMCHAR (2.x)

```
EM_LINEFROMCHAR  
wParam = (WPARAM) ich;    /* character index    */  
lParam = 0L;              /* not used, must be zero */
```

An application sends an EM\_LINEFROMCHAR message to retrieve the line number of the line that contains the specified character index. A character index is the number of characters from the beginning of the edit control.

This message is processed only by multiline edit controls.

Parameter	Description
ich	Value of wParam. Specifies the character index of the character contained in the line whose number is to be retrieved. If the ich parameter is -1, either the line number of the current line (the line containing the caret) is retrieved or, if there is a selection, the line number of the line containing the beginning of the selection is retrieved.

### Returns

The return value is the zero-based line number of the line containing the character index specified by ich.

### Example

This example sends an EM\_LINEFROMCHAR message to retrieve the line number of the current line in a multiline edit control:

```
SendMessage(hdlg, ID_MYEDITCONTROL,  
             EM_LINEFROMCHAR, -1, 0);
```

See Also

EM\_LINEINDEX

## EM\_LINEINDEX (2.x)

```
EM_LINEINDEX  
wParam = (WPARAM) line; /* line number */  
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_LINEINDEX message to retrieve the character index of a line within a multiline edit control. The character index is the number of characters from the beginning of the edit control to the specified line.

This message is processed only by multiline edit controls.

Parameter	Description
line	Value of wParam. Specifies the zero-based line number. A value of -1 specifies the current line number (the line that contains the caret).

### Returns

The return value is the character index of the line specified in the line parameter, or it is -1 if the specified line number is greater than the number of lines in the edit control.

### Example

This example uses the EM\_GETLINECOUNT message to retrieve the number of lines in an edit control and then uses EM\_LINEINDEX to retrieve the character index for the last line in the edit control:

```
WPARAM cLines, index;
```

```
cLines = (WPARAM) SendMessage(hdlg, ID_MYEDITCONTROL,  
    EM_GETLINECOUNT, 0, 0);  
index = (WPARAM) SendMessage(hdlg, ID_MYEDITCONTROL,  
    EM_LINEINDEX, cLines - 1, 0);
```

See Also

EM\_LINEFROMCHAR

## EM\_LINELENGTH (2.x)

```
EM_LINELENGTH  
wParam = (WPARAM) ich;    /* character index    */  
lParam = 0L;              /* not used, must be zero */
```

An application sends an EM\_LINELENGTH message to retrieve the length of a line in an edit control.

Parameter	Description
ich	Value of wParam. Specifies the character index of a character in the line whose length is to be retrieved when EM_LINELENGTH is sent to a multiline edit control. If this parameter is -1, the message returns the number of unselected characters on lines containing selected characters. For example, if the selection extended from the fourth character of one line through the eighth character from the end of the next line, the return value would be 10 (three characters on the first line and seven on the next). When EM_LINELENGTH is sent to a single-line edit control, this parameter is ignored.

### Returns

The return value is the length, in bytes, of the line specified by the ich parameter when an EM\_LINELENGTH message is sent to a multiline edit control. The return value is the length, in bytes, of the text in the edit control when an EM\_LINELENGTH message is sent to a single-line edit control.

### Comments

Use the EM\_LINEINDEX message to retrieve a character index for a given line number within a multiline edit control.

### Example

This example sends an EM\_LINEINDEX message to retrieve the length of the first line in a multiline edit control (or the entire text of a single-line edit control):

```
SendMessage(hdlg, ID_MYEDITCONTROL,  
            EM_LINELENGTH, 0, 0);
```

### See Also

EM\_GETLINE

## EM\_LINESCROLL (2.x)

EM\_LINESCROLL

```
wParam = 0; /* not used, must be zero */  
lParam = MAKELPARAM(dv, dh); /* lines and characters to scroll */
```

An application sends an EM\_LINESCROLL message to scroll the text of a multiline edit control.

This message is processed only by multiline edit controls.

Parameter	Description
dv	Value of the low-order word of lParam. Specifies the number of lines to scroll vertically.
dh	Value of the high-order word of lParam. Specifies the number of character positions to scroll horizontally. This value is ignored if the edit control has either the <u>ES_RIGHT</u> or <u>ES_CENTER</u> style.

### Returns

The return value is nonzero if the message is sent to a multiline edit control, or it is zero if the message is sent to a single-line edit control.

### Comments

The edit control does not scroll vertically past the last line of text in the edit control. If the current line plus the number of lines specified by the dv parameter exceeds the total number of lines in the edit control, the value is adjusted so that the last line of the edit control is scrolled to the top of the edit-control window.

The EM\_LINESCROLL message can be used to scroll horizontally past the last character of any line.

### Example

This example sends an EM\_LINESCROLL message to scroll the text in a multiline edit control vertically by five lines:

```
SendMessage(hdlg, ID_MYEDITCONTROL,  
EM_LINESCROLL, 0, MAKELONG(5, 0));
```

## EM\_REPLACESEL (2.x)

EM\_REPLACESEL

```
wParam = 0; /* not used, must be zero */  
lParam = (LPARAM) (LPCSTR) lpszReplace; /* address of new string */
```

An application sends an EM\_REPLACESEL message to replace the current selection in an edit control with the text specified by the lpszReplace parameter.

Parameter	Description
lpszReplace	Value of lParam. Points to a null-terminated string containing the replacement text.

Returns

This message does not return a value.

Comments

Use the EM\_REPLACESEL message when you want to replace only a portion of the text in an edit control. If you want to replace all of the text, use the WM\_SETTEXT message.

If there is no current selection, the replacement text is inserted at the current cursor location.

Example

This example sets the selection to the beginning of the edit control and inserts the string "C:\":

```
SendDlgItemMessage(hdlg, ID_MYEDITCONTROL,  
    EM_SETSEL, 0, MAKELONG(0, 0));  
SendDlgItemMessage(hdlg, ID_MYEDITCONTROL,  
    EM_REPLACESEL, 0, (LPARAM) ((LPSTR) "C:\\"));
```

See Also

EM\_GETSEL, EM\_SETSEL

## EM\_SETHANDLE (2.x)

### EM\_SETHANDLE

```
wParam = (WPARAM) (HLOCAL) hloc; /* handle of local memory object */
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_SETHANDLE message to set the handle to the local memory that will be used by a multiline edit control.

This message is processed only by multiline edit controls.

Parameter	Description
hloc	Value of wParam. Identifies the local memory. This handle must have been created by a previous call to the <u>LocalAlloc</u> function using the <u>LMEM_MOVEABLE</u> flag. The memory should contain a null-terminated string, or the first byte of the allocated memory should be set to zero.

### Returns

This message does not return a value.

### Comments

Before an application sets a new memory handle, it should send an EM\_GETHANDLE message to retrieve the handle to the current memory buffer and should free that memory by using the LocalFree function.

Sending an EM\_SETHANDLE message clears the undo buffer (EM\_CANUNDO returns zero) and the internal modification flag (EM\_GETMODIFY returns zero). The edit-control window is redrawn.

An application can send this message to a multiline edit control in a dialog box only if it has created the dialog box with the DS\_LOCALEDIT style flag set.

### Example

This example frees the current memory for the edit control, allocates new memory, and reads up to BUF\_SIZE bytes of a file into the allocated memory. It then sends an EM\_SETHANDLE message to set the handle of the edit control to the new memory, effectively placing up to BUF\_SIZE bytes of the file into the edit control.

```
#define BUF_SIZE 4 * 1024

HFILE hf;
OFSTRUCT of;
HLOCAL hlocOldMem, hlocNewMem;
PSTR pBuf;
int cbRead;

/* Get the handle to the old memory and free it. */

hlocOldMem = (HLOCAL) SendDlgItemMessage(hdlg,
    ID_MYEDITCONTROL, EM_GETHANDLE, 0, 0);
LocalFree(hlocOldMem);

/* Allocate new memory and read the file into it. */

hlocNewMem = LocalAlloc(LMEM_MOVEABLE, BUF_SIZE);
pBuf = LocalLock(hlocNewMem);
of.cBytes = sizeof(OFSTRUCT);
hf = OpenFile("test.txt", &of, OF_READ);
cbRead = lread(hf, pBuf, BUF_SIZE);
pBuf[cbRead] = '\0'; /* add terminating null character */
lclose(hf);

/* Adjust the buffer for the amount actually read in. */

LocalReAlloc(hlocNewMem, cbRead, 0);

/* Set the handle to the new buffer. */
```

```
LocalUnlock(hlocNewMem);  
SendDlgItemMessage(hdlg, ID_MYEDITCONTROL,  
EM_SETHANDLE, (WPARAM) hlocNewMem, 0);
```

See Also

EM\_GETHANDLE, LocalAlloc, LocalFree



## EM\_SETMODIFY (2.x)

### EM\_SETMODIFY

```
wParam = (WPARAM) (UINT) fModified; /* modification flag */  
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_SETMODIFY message to set or clear the modification flag for an edit control. The modification flag indicates whether the text within the edit control has been modified. It is automatically set whenever the user changes the text. An EM\_GETMODIFY message can be sent to retrieve the value of the modification flag.

Parameter	Description
fModified	Value of wParam. Specifies the new value for the modification flag. A value of TRUE indicates the text has been modified, and a value of FALSE indicates it has not been modified.

### Returns

This message does not return a value.

### Example

This example sends an EM\_SETMODIFY message to clear the modification flag:

```
SendMessage(hDlg, ID_MYEDITCONTROL, EM_SETMODIFY, FALSE, 0);
```

### See Also

EM\_GETMODIFY

## EM\_SETPASSWORDCHAR (3.0)

```
EM_SETPASSWORDCHAR  
wParam = (WPARAM) (UINT) ch; /* character to display */  
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_SETPASSWORDCHAR message to set or remove a password character displayed in an edit control when the user types text. When a password character is set, that character is displayed for each character the user types.

This message has no effect on a multiline edit control.

Parameter	Description
ch	Value of wParam. Specifies the character to be displayed in place of the character typed by the user. If the ch parameter is zero, the actual characters typed by the user are displayed.

### Returns

The return value is nonzero if the message is sent to an edit control.

### Comments

When the EM\_SETPASSWORDCHAR message is received by an edit control, the edit control redraws all visible characters by using the character specified by the ch parameter.

If the edit control is created with the ES\_PASSWORD style, the default password character is set to an asterisk (\*). This style is removed if an EM\_SETPASSWORDCHAR message is sent with the wParam parameter set to zero.

### Example

This example sends an EM\_SETPASSWORDCHAR message to set the password character of an edit control to a question mark:

```
SendMessage(hdlg, ID_MYEDITCONTROL,  
EM_SETPASSWORDCHAR, (WORD) '?', 0);
```

### See Also

EM\_GETPASSWORDCHAR

## EM\_SETREADONLY (3.1)

EM\_SETREADONLY

```
wParam = (WPARAM) (BOOL) fReadOnly; /* read-only flag */  
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_SETREADONLY message to set the read-only state of an edit control.

Parameter	Description
fReadOnly	Value of wParam. Specifies whether to set or remove the read-only state of the edit control. A value of TRUE sets the state to read-only; a value of FALSE sets the state to read/write.

### Returns

The return value is nonzero if the operation is successful, or it is zero if an error occurs.

### Comments

When the state of an edit control is set to read-only, the user cannot change the text within the edit control.

EM\_SETREADONLY does not have a corresponding message to retrieve the current style. Calling the GetWindowLong function with the GWL\_STYLE offset retrieves the full control style.

### Example

This example sets the state of an edit control to read-only:

```
SendMessage(hdlg, IDD_EDIT, EM_SETREADONLY,  
TRUE, 0L);
```

### See Also

GetWindowLong

## EM\_SETRECT (2.x)

### EM\_SETRECT

```
wParam = 0; /* not used, must be zero */  
lParam = (LPARAM) (const RECT FAR*) lprc; /* address of RECT */
```

An application sends an EM\_SETRECT message to set the formatting rectangle of a multiline edit control. The formatting rectangle is the limiting rectangle of the text. The limiting rectangle is independent of the size of the edit-control window. When the edit control is first created, the formatting rectangle is the same as the client area of the edit-control window. By using the EM\_SETRECT message, an application can make the formatting rectangle larger or smaller than the edit-control window.

This message is processed only by multiline edit controls.

Parameter	Description
lprc	Value of lParam. Points to a <u>RECT</u> structure that specifies the new dimensions of the rectangle.

### Returns

This message does not return a value.

### Comments

The EM\_SETRECT message causes the text of the edit control to be redrawn. To change the size of the formatting rectangle without redrawing the text, use the EM\_SETRECTNP message.

If the edit control does not have a horizontal scroll bar, and the formatting rectangle is set to be larger than the edit-control window, lines of text exceeding the width of the edit-control window (but smaller than the width of the formatting rectangle) are clipped instead of wrapped.

If the edit control contains a border, the formatting rectangle is reduced by the size of the border. If you are adjusting the rectangle returned by an EM\_GETRECT message, you must remove the size of the border before using the rectangle with the EM\_SETRECT message.

### Example

This example retrieves the current formatting rectangle for a multiline edit control, removes the border width dimensions, and sets the right border to 32767 so that all text sent to the edit control is clipped rather than wrapped if it exceeds the width of the edit-control window. The example then sends an EM\_SETRECT message to set the new formatting rectangle.

```
RECT rect;
```

```
SendDlgItemMessage(hdlg, ID_MYEDITCONTROL,  
    EM_GETRECT, 0, (LPARAM) (RECT FAR*) &rect);  
rect.left = 0; /* remove border width */  
rect.right = 32767; /* clip all lines */  
rect.bottom += rect.top; /* remove border height */  
rect.top = 0; /* remove border height */  
SendDlgItemMessage(hdlg, ID_MYEDITCONTROL,  
    EM_SETRECT, 0, (LPARAM) (RECT FAR*) &rect);
```

### See Also

EM\_GETRECT, EM\_SETRECTNP, RECT

## EM\_SETRECTNP (2.x)

EM\_SETRECTNP

```
wParam = 0; /* not used, must be zero */  
lParam = (LPARAM) (const RECT FAR*) lprc; /* address of RECT*/
```

An application sends an EM\_SETRECTNP message to set the formatting rectangle of a multiline edit control. The formatting rectangle is the limiting rectangle of the text. The limiting rectangle is independent of the size of the edit-control window. When the edit control is first created, the formatting rectangle is the same as the client area of the edit-control window. By using the EM\_SETRECTNP message, an application can make the formatting rectangle larger or smaller than the edit-control window.

The EM\_SETRECTNP message is identical to the EM\_SETRECT message, except that the edit-control window is not redrawn.

This message is processed only by multiline edit controls.

Parameter	Description
lprc	Value of lParam. Points to a <u>RECT</u> structure that specifies the new dimensions of the rectangle.

Returns

This message does not return a value.

See Also

EM\_GETRECT, EM\_SETRECT, RECT

## EM\_SETSEL (2.x)

```
EM_SETSEL
wParam = (WPARAM) (UINT) fScroll;    /* flag for caret scrolling */
lParam = MAKELPARAM(ichStart, ichEnd); /* start and end positions  */
/
```

An application sends an EM\_SETSEL message to select a range of characters in an edit control.

Parameter	Description
fScroll	Value of wParam. When this parameter is zero, the caret is scrolled into view. When this parameter is one, the caret is not scrolled into view.
ichStart	Value of the low-order word of lParam. Specifies the starting position.
ichEnd	Value of the high-order word of lParam. Specifies the ending position.

### Returns

The return value is nonzero if the message is sent to an edit control.

### Comments

If the ichStart parameter is 0 and the ichEnd parameter is -1, all the text in the edit control is selected. If ichStart is -1, any current selection is removed. The caret is placed at the end of the selection indicated by the greater of the two values ichEnd and ichStart.

### Example

This example sends an EM\_SETSEL message to select the entire contents of an edit control. It then sends a WM\_CUT message to copy the contents of the edit control to the clipboard and then to delete the contents of the edit control.

```
SendDlgItemMessage(hdlg, ID_MYEDITCONTROL,
    EM_SETSEL, 0, MAKELONG(0, -1));
SendDlgItemMessage(hdlg, ID_MYEDITCONTROL,
    WM_CUT, 0, 0);
```

### See Also

EM\_GETSEL, EM\_REPLACESEL

### Windows 3.1 changes

The meaning of the `wParam` parameter has changed. The `wParam` parameter specifies whether or not to scroll the caret.

## EM\_SETTABSTOPS (3.0)

EM\_SETTABSTOPS

```
wParam = (WPARAM) cTabs; /* number of tab stops */  
lParam = (LPARAM) (const int FAR*) lpTabs; /* tab-stop array */
```

An application sends an EM\_SETTABSTOPS message to set the tab stops in a multiline edit control (MLE). When text is copied to an MLE, any tab character in the text causes space to be generated up to the next tab stop.

This message is processed only by MLEs.

Parameter	Description
cTabs	Value of wParam. Specifies the number of tab stops contained in the lpTabs parameter. If this parameter is 0, the lpTabs parameter is ignored and default tab stops are set at every 32 dialog box units. If this parameter is 1, tab stops are set at every n dialog box units, where n is the distance pointed to by the lpTabs parameter. If the cTabs parameter is greater than 1, lpTabs points to an array of tab stops.
lpTabs	Low and high-order words of lParam. Points to an array of unsigned integers specifying the tab stops, in dialog box units. If the cTabs parameter is 1, lpTabs points to an unsigned integer containing the distance between all tab stops, in dialog units.

### Returns

The return value is nonzero if the tabs were set; otherwise, the return value is zero.

### Comments

The EM\_SETTABSTOPS message does not automatically redraw the edit-control window. If the application is changing the tab stops for text already in the edit control, it should call the [InvalidateRect](#) function to redraw the edit-control window.

### Example

This example sends an EM\_SETTABSTOPS message to set tab stops at every 64 dialog box units. It then calls [InvalidateRect](#) to redraw the edit-control window.

```
WORD wTabSpacing = 64;
```

```
SendMessage(hdlg, ID_MYEDITCONTROL,  
             EM_SETTABSTOPS, 1, (LPARAM) (int far*) &wTabSpacing);  
InvalidateRect(GetDlgItem(hdlg, ID_MYEDITCONTROL),  
                NULL, TRUE);
```

### See Also

[GetDialogBaseUnits](#)



## EM\_SETWORDBREAKPROC (3.1)

```
EM_SETWORDBREAKPROC  
wParam = 0; /* not used, must be zero */  
lParam = (LPARAM) (EDITWORDBREAKPROC) ewbprc; /* address of function */  
/
```

An application sends the EM\_SETWORDBREAKPROC message to an edit control to replace the default wordwrap function with an application-defined wordwrap function.

Parameter	Description
ewbprc	Value of lParam. Specifies the procedure-instance address of the application-defined wordwrap function. The <u>MakeProcInstance</u> function must be used to create the address. For more information, see the <u>description of the WordBreakProc</u> callback function.

### Returns

This message does not return a value.

### Comments

A wordwrap function scans a text buffer (which contains text to be sent to the display), looking for the first word that does not fit on the current display line. The wordwrap function places this word at the beginning of the next line on the display.

A wordwrap function defines the point at which Windows should break a line of text for multiline edit controls, usually at a space character that separates two words. Either a multiline or a single-line edit control might call this function when the user presses arrow keys in combination with the CTRL key to move the cursor to the next word or previous word. The default wordwrap function breaks a line of text at a space character. The application-defined function may define wordwrap to occur at a hyphen or a character other than the space character.

### See Also

EM\_FMTLINES, EM\_GETWORDBREAKPROC, MakeProcInstance, WordBreakProc

## EM\_UNDO (2.x)

```
EM_UNDO  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends an EM\_UNDO message to undo the last edit-control operation.

### Parameters

This message has no parameters.

### Returns

The return value is always nonzero for a single-line edit control. For a multiline edit control, the return value is nonzero if the undo operation is successful or zero if the undo operation fails.

### Comments

An undo operation can also be undone. For example, you can restore deleted text with the first EM\_UNDO message and remove the text again with a second EM\_UNDO message as long as there is no intervening edit-control operation.

### Example

This example undoes the last edit-control operation:

```
SendMessage(hdlg, ID_MYEDITCONTROL, EM_UNDO, 0, 0);
```

### See Also

EM\_CANUNDO, EM\_EMPTYUNDOBUFFER, WM\_UNDO

## LB\_ADDSTRING (2.x)

LB\_ADDSTRING

```
wParam = 0; /* not used, must be zero */  
lParam = (LPARAM) (LPCSTR) lpsz; /* address of string to add */
```

An application sends an LB\_ADDSTRING message to add a string to a list box. If the list box does not have the CBS\_SORT style, the string is added to the end of the list. Otherwise, the string is inserted into the list and the list is sorted.

Parameter	Description
lpsz	Value of lParam. Points to the null-terminated string that is to be added. If the list box was created with an owner-drawn style but without the LBS_HASSTRINGS style, the value of the lpsz parameter is stored rather than the string it would otherwise point to.

### Returns

The return value is the zero-based index to the string in the list box. The return value is LB\_ERR if an error occurs; the return value is LB\_ERRSPACE if insufficient space is available to store the new string.

### Comments

If an owner-drawn list box was created with the LBS\_SORT style but not the LBS\_HASSTRINGS style, the WM\_COMPAREITEM message is sent one or more times to the owner of the list box so the new item can be properly placed in the list box.

### Example

This example adds the string "my string" to a list box:

```
DWORD dwIndex;
```

```
dwIndex = SendDlgItemMessage(hdlg, ID_MYLISTBOX,  
    LB_ADDSTRING, 0, (LPARAM) ((LPSTR) "my string"));
```

### See Also

LB\_DIR, LB\_INSERTSTRING, WM\_COMPAREITEM

## LB\_DELETESTRING (2.x)

LB\_DELETESTRING

```
wParam = (WPARAM) index; /* index of string to delete */  
lParam = 0L; /* not used, must be zero */
```

An application sends an LB\_DELETESTRING message to delete a string in a list box.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the string to delete.

### Returns

The return value is a count of the strings remaining in the list. The return value is LB\_ERR if the index parameter specifies an index greater than the number of items in the list.

### Comments

If the list box was created with an owner-drawn style but without the LBS\_HASSTRINGS style, a WM\_DELETEITEM message is sent to the owner of the list box so that the application can free any additional data associated with the item.

### Example

This example deletes the first string in a list box:

```
DWORD dwRemaining;
```

```
dwRemaining = SendDlgItemMessage(hDlg, ID_MYLISTBOX,  
    LB_DELETESTRING, 0, 0);
```

### See Also

LB\_RESETCONTENT, WM\_DELETEITEM

## LB\_DIR (2.x)

```
LB_DIR
wParam = (WPARAM) (UINT) uAttrs;    /* file attributes */
lParam = (LPARAM) (LPCSTR) lpszFileSpec; /* filename string's address */
/
```

An application sends an LB\_DIR message to add a list of filenames to a list box.

Parameter	Description																		
uAttrs	Value of wParam. Specifies the attributes of the files to be added to the list box. It can be any combination of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>DDL_READWRITE</td><td>File can be read from or written to.</td></tr><tr><td>DDL_READONLY</td><td>File can be read from but not written to.</td></tr><tr><td>DDL_HIDDEN</td><td>File is hidden and does not appear in a directory listing.</td></tr><tr><td>DDL_SYSTEM</td><td>File is a system file.</td></tr><tr><td>DDL_DIRECTORY</td><td>The name pointed to by the lpszFileSpec parameter specifies a directory.</td></tr><tr><td>DDL_ARCHIVE</td><td>File has been archived.</td></tr><tr><td>DDL_DRIVES</td><td>All drives that match the name specified by the lpszFileSpec parameter are included. If the <u>DDL_DRIVES</u> flag is set, the <u>DDL_EXCLUSIVE</u> flag is set automatically. Therefore, to create a directory listing that includes drives and files, the developer must send this message twice: once with the DDL_DRIVES flag set and once with the flags for the rest of the list.</td></tr><tr><td>DDL_EXCLUSIVE</td><td>Exclusive flag. If the exclusive flag is set, only files of the specified type are listed. Otherwise, files of the specified type are listed in addition to files that do not match the specified type.</td></tr></tbody></table>	Value	Meaning	DDL_READWRITE	File can be read from or written to.	DDL_READONLY	File can be read from but not written to.	DDL_HIDDEN	File is hidden and does not appear in a directory listing.	DDL_SYSTEM	File is a system file.	DDL_DIRECTORY	The name pointed to by the lpszFileSpec parameter specifies a directory.	DDL_ARCHIVE	File has been archived.	DDL_DRIVES	All drives that match the name specified by the lpszFileSpec parameter are included. If the <u>DDL_DRIVES</u> flag is set, the <u>DDL_EXCLUSIVE</u> flag is set automatically. Therefore, to create a directory listing that includes drives and files, the developer must send this message twice: once with the DDL_DRIVES flag set and once with the flags for the rest of the list.	DDL_EXCLUSIVE	Exclusive flag. If the exclusive flag is set, only files of the specified type are listed. Otherwise, files of the specified type are listed in addition to files that do not match the specified type.
Value	Meaning																		
DDL_READWRITE	File can be read from or written to.																		
DDL_READONLY	File can be read from but not written to.																		
DDL_HIDDEN	File is hidden and does not appear in a directory listing.																		
DDL_SYSTEM	File is a system file.																		
DDL_DIRECTORY	The name pointed to by the lpszFileSpec parameter specifies a directory.																		
DDL_ARCHIVE	File has been archived.																		
DDL_DRIVES	All drives that match the name specified by the lpszFileSpec parameter are included. If the <u>DDL_DRIVES</u> flag is set, the <u>DDL_EXCLUSIVE</u> flag is set automatically. Therefore, to create a directory listing that includes drives and files, the developer must send this message twice: once with the DDL_DRIVES flag set and once with the flags for the rest of the list.																		
DDL_EXCLUSIVE	Exclusive flag. If the exclusive flag is set, only files of the specified type are listed. Otherwise, files of the specified type are listed in addition to files that do not match the specified type.																		
lpszFileSpec	Value of lParam. Points to the null-terminated string that specifies the filename to add to the list. If the filename contains wildcards (for example, *.*), all files that match and have the attributes specified by the uAttrs parameter are added to the list.																		

### Returns

The return value is the zero-based index of the last filename added to the list. The return value is LB\_ERR if an error occurs; the return value is LB\_ERRSPACE if insufficient space is available to store the new strings.

### Example

This example adds the names of all available drives to a list box:

```
DWORD dwIndexLastItem;
```

```
dwIndexLastItem = SendDlgItemMessage(hdlg, ID_MYLISTBOX, LB_DIR,  
0x4000 | 0x8000, (LPARAM) ((LPSTR) ""));
```

### See Also

DlgDirList, LB\_ADDSTRING, LB\_INSERTSTRING

## LB\_FINDSTRING (3.0)

LB\_FINDSTRING

```
wParam = (WPARAM) indexStart; /* item before start of search */  
lParam = (LPARAM) (LPCSTR) lpszFind; /* address of search string */
```

An application sends an LB\_FINDSTRING message to search a list box for an item that begins with the characters in a specified string.

Parameter	Description
indexStart	Value of wParam. Specifies the zero-based index of the item before the first item to be searched. When the search reaches the bottom of the list box, it continues from the top of the list box back to the item specified by the indexStart parameter. If indexStart is -1, the entire list box is searched from the beginning.
lpszFind	Value of lParam. Points to the null-terminated string that contains the prefix to search for. The search is not case-sensitive, so this string can contain any combination of uppercase and lowercase letters.

Returns

The return value is the index of the matching item, or it is LB\_ERR if the search was unsuccessful.

Comments

If the combo box was created with an owner-drawn style but without the CBS\_HASSTRINGS style, the action taken by the LB\_FINDSTRING message depends on whether the CBS\_SORT style is used. If the CBS\_SORT style is used, WM\_COMPAREITEM messages are sent to the owner of the combo box to determine which item matches the specified string. Otherwise, LB\_FINDSTRING attempts to match the doubleword value against the value of the lpszFind parameter.

Example

This example searches for the string "my string" in a list box and copies it, if found, to the szBuf buffer:

```
char szBuf[20];  
DWORD dwIndex;  
  
dwIndex = SendDlgItemMessage(hdlg, ID_MYLISTBOX,  
    LB_FINDSTRING, 0, (LPARAM) ((LPSTR) "my string"));  
if (dwIndex != LB_ERR)  
    SendDlgItemMessage(hdlg, ID_MYLISTBOX,  
        LB_GETTEXT, (WPARAM) dwIndex, (LPARAM) ((LPSTR) szBuf));
```

See Also

LB\_FINDSTRINGEXACT, LB\_SELECTSTRING, WM\_COMPAREITEM

## LB\_FINDSTRINGEXACT (3.1)

LB\_FINDSTRINGEXACT

```
wParam = (WPARAM) indexStart; /* item before start of search */  
lParam = (LPARAM) (LPCSTR) lpszFind; /* address of search string */
```

An application sends an LB\_FINDSTRINGEXACT message to find the first list box string that matches the string specified in the lpszFind parameter.

Parameter	Description
indexStart	Value of wParam. Specifies the zero-based index of the item before the first item to be searched. When the search reaches the bottom of the list box, it continues from the top of the list box back to the item specified by the indexStart parameter. If indexStart is -1, the entire list box is searched from the beginning.
lpszFind	Value of lParam. Points to the null-terminated string to search for. This string can contain a complete filename, including the extension. The search is not case-sensitive, so the string can contain any combination of uppercase and lowercase letters.

### Returns

The return value is the index of the matching item, or it is LB\_ERR if the search was unsuccessful.

### Comments

If the combo box was created with an owner-drawn style but without the CBS\_HASSTRINGS style, the action taken by the LB\_FINDSTRINGEXACT message depends on whether the CBS\_SORT style is used. If the CBS\_SORT style is used, WM\_COMPAREITEM messages are sent to the owner of the combo box to determine which item matches the specified string. Otherwise, LB\_FINDSTRINGEXACT attempts to match the doubleword value against the value of the lpszFind parameter.

### Example

This example searches for the string "my string" in a list box and copies it, if found, to the szBuf buffer:

```
char szBuf[20];  
DWORD dwIndex;  
  
dwIndex = SendDlgItemMessage(hdlg, ID_MYLISTBOX,  
    LB_FINDSTRINGEXACT, 0, (LPARAM) ((LPSTR) "my string"));  
if (dwIndex != LB_ERR)  
    SendDlgItemMessage(hdlg, ID_MYLISTBOX,  
        LB_GETTEXT, (WPARAM) dwIndex, (LPARAM) ((LPSTR) szBuf));
```

### See Also

LB\_FINDSTRING, LB\_SELECTSTRING, WM\_COMPAREITEM

## LB\_GETCARETINDEX (3.1)

```
LB_GETCARETINDEX  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends an LB\_GETCARETINDEX message to determine the index of the item that has the focus rectangle in a multiple-selection list box. The item may or may not be selected.

### Parameters

This message has no parameters.

### Returns

The return value is the zero-based index of the item that has the focus rectangle in a list box. If the list box is a single-selection list box, the return value is the index of the item that is selected, if any.

### Example

This example sends an LB\_GETCARETINDEX message to retrieve the index of the item that has the focus rectangle in the list box:

```
LRESULT lrIndex;
```

```
lrIndex = SendDlgItemMessage(hdlg, ID_MYLISTBOX,  
    LB_GETCARETINDEX, 0, 0L);
```

See Also

[LB\\_SETCARETINDEX](#)



## LB\_GETCOUNT (2.x)

```
LB_GETCOUNT  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends an LB\_GETCOUNT message to retrieve the number of items in a list box.

### Parameters

This message has no parameters.

### Returns

The return value is the number of items in the list box, or it is LB\_ERR if an error occurs.

### Comments

The returned count is one greater than the index value of the last item (the index is zero-based).

### Example

This example retrieves the number of items in a list box:

```
DWORD cListItems;
```

```
cListItems = SendDlgItemMessage(hdlg, ID_MYLISTBOX, LB_GETCOUNT, 0, 0)  
;
```

## LB\_GETCURSEL (2.x)

```
LB_GETCURSEL  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends an LB\_GETCURSEL message to retrieve the index of the currently selected item, if any, in a single-selection list box.

### Parameters

This message has no parameters.

### Returns

The return value is the zero-based index of the currently selected item. It is LB\_ERR if no item is currently selected.

### Comments

An application should use the LB\_GETCARETINDEX to retrieve the index of the item that has the focus rectangle in a multiple-selection list box.

The LB\_GETCURSEL message cannot be sent to a multiple-selection list box.

### Example

This example retrieves the index of the currently selected string in a list box and then retrieves that string:

```
char szBuf[20];  
DWORD dwIndex;  
  
dwIndex = SendDlgItemMessage(hdlg, ID_MYLISTBOX, LB_GETCURSEL, 0, 0);  
if (dwIndex != LB_ERR)  
    SendDlgItemMessage(hdlg, ID_MYLISTBOX,  
        LB_GETTEXT, (WPARAM) dwIndex, (LPARAM) ((LPSTR) szBuf));
```

### See Also

LB\_GETSEL, LB\_SETCURSEL, LB\_SELECTSTRING

## LB\_GETHORIZONTALAEXTENT (3.0)

```
LB_GETHORIZONTALAEXTENT  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends the LB\_GETHORIZONTALAEXTENT message to retrieve from a list box the width, in pixels, by which the list box can be scrolled horizontally if the list box has a horizontal scroll bar.

### Parameters

This message has no parameters.

### Returns

The return value is the scrollable width of the list box, in pixels.

### Comments

To respond to the LB\_GETHORIZONTALAEXTENT message, the list box must have been defined with the WS\_HSCROLL style.

This message is not useful for multicolumn listboxes.

### Example

This example gets the horizontal extent of a list box:

```
SendDlgItemMessage(hDlg, ID_MYLISTBOX,  
LB_GETHORIZONTALAEXTENT, 0, 0L);
```

### See Also

LB\_SETHORIZONTALAEXTENT

## LB\_GETITEMDATA (3.0)

```
LB_GETITEMDATA
wParam = (WPARAM) index;    /* item index */
lParam = 0L; /* not used, must be zero */
```

An application sends the LB\_GETITEMDATA message to retrieve the application-supplied doubleword value associated with the specified item in a list box. (This is the value of the lParam parameter of an LB\_SETITEMDATA message.)

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the item.

### Returns

The return value is the doubleword value associated with the item, or it is LB\_ERR if an error occurs.

### Example

This example retrieves the value associated with an item in a list box. The value is the handle of a global memory object.

```
HGLOBAL hglbData;
LPSTR lpLBData;
HWND hListBox;
WPARAM nIndex;

if ((hglbData = (HGLOBAL) LOWORD(SendMessage(hListBox, LB_GETITEMDATA,
nIndex, 0)))) {
    if ((lpLBData = GlobalLock(hglbData))) {
        . /* Access or manipulate the data */
        .
        GlobalUnlock(hglbData);
    }
}
```

### See Also

LB\_SETITEMDATA

## LB\_GETITEMHEIGHT (3.1)

```
LB_GETITEMHEIGHT  
wParam = (WPARAM) index;    /* item index    */  
lParam = 0L; /* not used, must be zero */
```

An application sends an LB\_GETITEMHEIGHT message to determine the height of items in a list box.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the item in the list box. This parameter is used only if the list box has the <u>LBS_OWNERDRAWVARIABLE</u> style; otherwise, it should be set to zero.

### Returns

The return value is the height, in pixels, of the items in the list box. The return value is the height of the item specified by the index parameter if the list box has the LBS\_OWNERDRAWVARIABLE style. The return value is LB\_ERR if an error occurs.

### Example

This example sends an LB\_GETITEMHEIGHT message to retrieve the height of the items in a list box:

```
LRESULT lrHeight;
```

```
lrHeight = SendDlgItemMessage(hdlg, ID_MYLISTBOX,  
    LB_GETITEMHEIGHT, 0, 0L);
```

### See Also

LB\_GETITEMRECT, LB\_SETITEMHEIGHT, WM\_MEASUREITEM

## LB\_GETITEMRECT (3.0)

LB\_GETITEMRECT

```
wParam = (WPARAM) index; /* item index */
```

```
lParam = (LPARAM) (RECT FAR*) lprc; /* address of RECT structure */
```

An application sends an LB\_GETITEMRECT message to retrieve the dimensions of the rectangle that bounds an item as it is currently displayed in the list box window.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the item.
lprc	Value of lParam. Specifies a long pointer to a <u>RECT</u> structure that receives the client coordinates for the item in the list box.

### Returns

The return value is LB\_ERR if an error occurs.

### See Also

LB\_GETITEMHEIGHT, LB\_SETITEMHEIGHT, WM\_MEASUREITEM, RECT

## LB\_GETSEL (2.x)

```
LB_GETSEL  
wParam = (WPARAM) index;    /* item index    */  
lParam = 0L; /* not used, must be zero */
```

An application sends an LB\_GETSEL message to retrieve the selection state of an item.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the item.

### Returns

The return value is a positive number if an item is selected; otherwise, it is zero. The return value is LB\_ERR if an error occurs.

### See Also

[LB\\_GETCURSEL](#), [LB\\_SELECTSTRING](#), [LB\\_SELITEMRANGE](#), [LB\\_SETSEL](#)

## LB\_GETSELCOUNT (3.0)

```
LB_GETSELCOUNT  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends an LB\_GETSELCOUNT message to retrieve the total number of selected items in a multiple-selection list box.

### Parameters

This message has no parameters.

### Returns

The return value is the count of selected items in a list box. The return value is LB\_ERR if the list box is a single-selection list box.

### See Also

[LB\\_GETSELITEMS](#)



## LB\_GETSELITEMS (3.0)

### LB\_GETSELITEMS

```
wParam = (WPARAM) cItems; /* maximum number of items */  
lParam = (LPARAM) (int FAR*) lpItems; /* address of buffer */
```

An application sends an LB\_GETSELITEMS message to fill a buffer with an array of integers that specify the item numbers of selected items in a multiple-selection list box.

Parameter	Description
cItems	Value of wParam. Specifies the maximum number of selected items whose item numbers are to be placed in the buffer.
lpItems	Value of lParam. Specifies a long pointer to a buffer large enough for the number of integers specified by the cItems parameter.

### Returns

The return value is the actual number of items placed in the buffer. The return value is LB\_ERR if the list box is a single-selection list box.

See Also

[LB\\_GETSELCOUNT](#)

## LB\_GETTEXT (2.x)

### LB\_GETTEXT

```
wParam = (WPARAM) index; /* item index */
lParam = (LPARAM) (LPCSTR) lpszBuffer; /* address of buffer */
```

An application sends an LB\_GETTEXT message to retrieve a string from a list box.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the string to retrieve.
lpszBuffer	Value of lParam. Points to the buffer that receives the string. The buffer must have sufficient space for the string and a terminating null character. An <u>LB_GETTEXTLEN</u> message can be sent before the LB_GETTEXT message to retrieve the length, in bytes, of the string.

### Returns

The return value is the length of the string, in bytes, excluding the terminating null character. The return value is LB\_ERR if the index parameter does not specify a valid index.

### Comments

If the list box was created with an owner-drawn style but without the LBS\_HASSTRINGS style, the buffer pointed to by the lpszBuffer parameter receives the doubleword value associated with the item.

### Example

This example retrieves the length of the first item in the list box, allocates sufficient memory for the string, and then sends an LB\_GETTEXT message to retrieve the string:

```
DWORD cbItemString;
PSTR psz;

cbItemString = SendDlgItemMessage(hdlg, ID_MYLISTBOX,
    LB_GETTEXTLEN, 0, 0);
if (cbItemString != LB_ERR) {
    psz = (PSTR) LocalAlloc(LMEM_FIXED, (WORD) cbItemString);
    SendDlgItemMessage(hdlg, ID_MYLISTBOX,
        LB_GETTEXT, 0, (LPARAM) ((LPSTR) psz));
}
```

### See Also

LB\_GETTEXTLEN

## LB\_GETTEXTLEN (2.x)

LB\_GETTEXTLEN

```
wParam = (WPARAM) index;    /* item index */  
lParam = 0L; /* not used, must be zero */
```

An application sends an LB\_GETTEXTLEN message to retrieve the length of a string in a list box.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the string.

### Returns

The return value is the length of the string, in bytes, excluding the terminating null character. The return value is LB\_ERR if the index parameter does not specify a valid index.

### Example

This example retrieves the length of the first item in the list box:

```
DWORD cbItemString;
```

```
cbItemString = SendDlgItemMessage(hdlg, ID_MYLISTBOX,  
    LB_GETTEXTLEN, 0, 0);
```

See Also

LB\_GETTEXT

## LB\_GETTOPINDEX (3.0)

```
LB_GETTOPINDEX  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends an LB\_GETTOPINDEX message to retrieve the index of the first visible item in a list box. Initially, the item with index 0 is at the top of the list box, but if the list box is scrolled, another item may be at the top.

### Parameters

This message has no parameters.

### Returns

The return value is the zero-based index of the first visible item in a list box.

### See Also

[LB\\_SETTOPINDEX](#)

## LB\_INSERTSTRING (2.x)

```
LB_INSERTSTRING  
wParam = (WPARAM) index; /* item index */  
lParam = (LPARAM) (LPCSTR) lpsz; /* address of string to insert */
```

An application sends an LB\_INSERTSTRING message to insert a string into a list box. Unlike the LB\_ADDSTRING message, the LB\_INSERTSTRING message does not cause a list with the LBS\_SORT style to be sorted.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the position at which to insert the string. If this parameter is -1, the string is added to the end of the list.
lpsz	Value of lParam. Points to the null-terminated string that is to be inserted. If the list was created with an owner-drawn style but without the <u>LBS_HASSTRINGS</u> style, the value of the lpsz parameter is stored rather than the string it would otherwise point to.

### Returns

The return value is the index of the position at which the string was inserted. The return value is LB\_ERR if an error occurs. The return value is LB\_ERRSPACE if insufficient space is available to store the new string.

### Example

This example inserts the string "my string" into the third position of the list box:

```
SendMessage(hdlg, ID_MYLISTBOX,  
             LB_INSERTSTRING, 2, (LPARAM) ((LPSTR) "my string"));
```

### See Also

LB\_ADDSTRING, LB\_DIR

## LB\_RESETCONTENT (2.x)

```
LB_RESETCONTENT  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends an LB\_RESETCONTENT message to remove all items from a list box.

### Parameters

This message has no parameters.

### Returns

This message does not return a value.

### Comments

If the list box was created with an owner-drawn style but without the LBS\_HASSTRINGS style, the owner of the list box receives a WM\_DELETEITEM message for each item in the list box.

### Example

This example removes all items from a list box:

```
SendDlgItemMessage(hdlg, ID_MYLISTBOX, LB_RESETCONTENT, 0, 0);
```

### See Also

LB\_DELETESTRING, WM\_DELETEITEM

## LB\_SELECTSTRING (2.x)

LB\_SELECTSTRING

```
wParam = (WPARAM) indexStart; /* item before start of search */  
lParam = (LPARAM) (LPCSTR) lpszFind; /* address of search string */
```

An application sends an LB\_SELECTSTRING message to search a list box for an item that begins with the characters in a specified string. If a matching item is found, the item is selected.

Parameter	Description
indexStart	Value of wParam. Specifies the zero-based index of the item before the first item to be searched. When the search reaches the bottom of the list box, it continues from the top of the list box back to the item specified by the indexStart parameter. If indexStart is -1, the entire list box is searched from the beginning.
lpszFind	Value of lParam. Points to the null-terminated string that contains the prefix to search for. The search is not case-sensitive, so this string can contain any combination of uppercase and lowercase letters.

### Returns

The return value is the index of the selected item if the search was successful. The return value is LB\_ERR if the search was unsuccessful and the current selection is not changed.

### Comments

The list box is scrolled, if necessary, to bring the selected item into view.

An item is selected only if its initial characters (from the starting point) match the characters in the string specified by the lpszFind parameter.

If the combo box was created with an owner-drawn style but without the CBS\_HASSTRINGS style, the action taken by the LB\_SELECTSTRING message depends on whether the CBS\_SORT style is used. If the CBS\_SORT style is used, WM\_COMPAREITEM messages are sent to the owner of the combo box to determine which item matches the specified string. Otherwise, LB\_SELECTSTRING attempts to match the doubleword value against the value of the lpszFind parameter.

### Example

This example searches the entire list box for an item that matches the string "my string" and, if the item is found, selects it:

```
DWORD dwIndexFoundString;
```

```
dwIndexFoundString = SendDlgItemMessage(hdlg, ID_MYLISTBOX,  
    LB_SELECTSTRING, -1, (LPARAM) ((LPSTR) "my string"));
```

### See Also

[LB\\_FINDSTRING](#), [LB\\_FINDSTRINGEXACT](#), [LB\\_SELITEMRANGE](#), [LB\\_SETCURSEL](#),  
[LB\\_SETSEL](#), [WM\\_COMPAREITEM](#)

## LB\_SELITEMRANGE (3.0)

### LB\_SELITEMRANGE

```
wParam = (WPARAM) (BOOL) fSelect; /* selection flag */  
lParam = MAKELPARAM(wFirst, wLast); /* first and last items */
```

An application sends an LB\_SELITEMRANGE message to select one or more consecutive items in a multiple-selection list box.

Parameter	Description
fSelect	Value of wParam. Specifies how to set the selection. If the fSelect parameter is nonzero, the string is selected and highlighted; if fSelect is zero, the highlight is removed and the string is no longer selected.
wFirst	Value of the low-order word of lParam. Specifies the zero-based index of the first item to set.
wLast	Value of the high-order word of lParam. Specifies the zero-based index of the last item to set.

### Returns

The return value is LB\_ERR if an error occurs.

### Comments

This message should be used only with multiple-selection list boxes.

### See Also

LB\_SELECTSTRING, LB\_SETSEL



## ■ LB\_SETCARETINDEX (3.1)

```
LB_SETCARETINDEX  
wParam = (WPARAM) index;    /* item index    */  
lParam = MAKELPARAM(fScroll, 0); /* flag for scrolling item */
```

An application sends an LB\_SETCARETINDEX message to set the focus rectangle to the item at the specified index in a multiple-selection list box. If the item is not visible, it is scrolled into view.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the item to receive the focus rectangle in the list box.
fScroll	Value of lParam. If this value is zero, the item is scrolled until it is fully visible. If this value is nonzero, the item is scrolled until it is at least partially visible.

### Returns

The return value is LB\_ERR if an error occurs.

### Example

This example sends an LB\_SETCARETINDEX message to set the focus rectangle to an item in a list box:

```
WPARAM wIndex;
```

```
wIndex = 0; /* set index to first item */
```

```
SendDlgItemMessage(hdlg, ID_MYLISTBOX, LB_SETCARETINDEX,  
wIndex, 0L);
```

See Also

LB\_GETCARETINDEX

## Windows 3.1 Changes

In previous versions of Windows, the lParam was not used.

## LB\_SETCOLUMNWIDTH (3.0)

LB\_SETCOLUMNWIDTH

```
wParam = (WPARAM) cxColumn; /* column width */  
lParam = 0L; /* not used, must be zero */
```

An application sends an LB\_SETCOLUMNWIDTH message to a multiple-column list box (created with the LBS\_MULTICOLUMN style) to set the width, in pixels, of all columns in the list box.

Parameter	Description
cxColumn	Value of wParam. Specifies the width, in pixels, of all columns.

Returns

This message does not return a value.

Example

This example sets the width of the columns in a multiple-column list box:

```
WPARAM wColWidth;
```

```
wColWidth = 100; /* set column width to 100 pixels */
```

```
SendDlgItemMessage(hDlg, ID_MYLISTBOX, LB_SETCOLUMNWIDTH,  
wColWidth, 0L);
```

See Also

LB\_SETHORIZONTALEXTENT

## LB\_SETCURSEL (2.x)

```
LB_SETCURSEL  
wParam = (WPARAM) index;    /* item index    */  
lParam = 0L; /* not used, must be zero */
```

An application sends an LB\_SETCURSEL message to select a string and scroll it into view, if necessary. When the new string is selected, the list box removes the highlight from the previously selected string.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the string that is selected. If the index parameter is -1, the list box is set to have no selection.

### Returns

The return value is LB\_ERR if an error occurs. The return value will be LB\_ERR even though no error has occurred if the index parameter is -1.

### Comments

This message should be used only with single-selection list boxes. It cannot be used to set or remove a selection in a multiple-selection list box.

### See Also

LB\_GETCURSEL, LB\_SELECTSTRING, LB\_SETSEL

## LB\_SETHORIZONTALEXTENT (3.0)

LB\_SETHORIZONTALEXTENT

```
wParam = (WPARAM) cxExtent; /* horizontal scroll width */  
lParam = 0L; /* not used, must be zero */
```

An application sends the LB\_SETHORIZONTALEXTENT message to set the width, in pixels, by which a list box can be scrolled horizontally. If the size of the list box is smaller than this value, the horizontal scroll bar horizontally scrolls items in the list box. If the size of the list box is equal to or greater than this value, the horizontal scroll bar is hidden.

Parameter	Description
cxExtent	Value of wParam. Specifies the number of pixels by which the list box can be scrolled.

Returns

This message does not return a value.

Comments

To respond to the LB\_SETHORIZONTALEXTENT message, the list box must have been defined with the WS\_HSCROLL style.

By default, the horizontal extent of a list box is zero. Windows does not display the scroll bar unless the horizontal extent is set to a value greater than the width, in pixels, of the client area of the list box.

This message is not useful for multicolumn listboxes. Multicolumn listboxes should instead use the LB\_SETCOLUMNWIDTH message.

Example

This example sets the horizontal extent of a list box based on the width of the string about to be added to the list box. The horizontal extent is set if the string is wider than the widest string in the list box and is wider than the client area of the list box.

```
DWORD dwStringExt;  
HDC hdcLB;  
PSTR pszString;  
TEXTMETRIC tm;  
WORD wLongest;  
WORD wLBWidth;
```

```
dwStringExt = GetTextExtent(hdcLB, (LPSTR) pszString,  
    strlen(pszString)) + tm.tmAveCharWidth;
```

```
if ((LOWORD(dwStringExt) > wLongest) &&  
    (LOWORD(dwStringExt) > wLBWidth)) {  
    SendDlgItemMessage(hDlg, ID_MYLISTBOX, LB_SETHORIZONTALEXTENT,  
        LOWORD(dwStringExt), 0L);  
    wLongest = LOWORD(dwStringExt);  
}
```

```
SendDlgItemMessage(hDlg, ID_MYLISTBOX, LB_ADDSTRING, 0,  
    (LPARAM) ((LPCSTR) pszString));
```

See Also

LB\_GETHORIZONTALEXTENT, LB\_SETCOLUMNWIDTH

## LB\_SETITEMDATA (3.0)

```
LB_SETITEMDATA
wParam = (WPARAM) index;    /* item index    */
lParam = (LPARAM) dwData;   /* value to associate with item */
```

An application sends the LB\_SETITEMDATA message to set a doubleword value associated with the specified item in a list box.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the item.
dwData	Value of lParam. Specifies the value to be associated with the item.

### Returns

The return value is LB\_ERR if an error occurs.

### Example

This example associates a handle of a 64-byte memory object with each item in a list box:

```
HGLOBAL hglbData;
LPSTR lpLBData;
HWND hListBox;
WPARAM nIndex;

case WM_INITDIALOG:
    if ((hglbData = GlobalAlloc(GMEM_MOVEABLE, 64))) {
        if ((lpLBData = GlobalLock(hglbData))) {
            . /* Store the data in the memory object. */
            .
            GlobalUnlock(hglbData);
        }
        SendMessage(hListBox, LB_SETITEMDATA, nIndex,
            MAKELONG(hglbData, 0));
    }
```

See Also

LB\_GETITEMDATA

## LB\_SETITEMHEIGHT (3.1)

LB\_SETITEMHEIGHT

```
wParam = (WPARAM) index; /* item index */  
lParam = MAKELPARAM(cyItem, 0); /* item height */
```

An application sends an LB\_SETITEMHEIGHT message to set the height of items in a list box. If the list box has the LBS\_OWNERDRAWVARIABLE style, this message sets the height of the item specified by the wParam parameter. Otherwise, this message sets the height of all items in the list box.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the item in the list box. This parameter is used only if the list box has the <u>LBS_OWNERDRAWVARIABLE</u> style; otherwise, it should be set to zero.
cyItem	Value of the low-order word of lParam. Specifies the height, in pixels, of the item.

### Returns

The return value is LB\_ERR if the index or height is invalid.

### Example

This example sends an LB\_SETITEMHEIGHT message to set the height of the items in a list box:

```
LPARAM lpmHeight;
```

```
SendDlgItemMessage(hdlg, ID_MYLISTBOX, LB_SETITEMHEIGHT,  
0, lpmHeight);
```

### See Also

LB\_GETITEMHEIGHT, LB\_GETITEMRECT, WM\_MEASUREITEM

## LB\_SETSEL (2.x)

### LB\_SETSEL

```
wParam = (WPARAM) (BOOL) fSelect; /* selection flag */  
lParam = MAKELPARAM(index, 0); /* item index*/
```

An application sends an LB\_SETSEL message to select a string in a multiple-selection list box.

Parameter	Description
fSelect	Value of wParam. Specifies how to set the selection. If the fSelect parameter is TRUE, the string is selected and highlighted; if fSelect is FALSE, the highlight is removed and the string is no longer selected.
index	Value of the low-order word of lParam. Specifies the zero-based index of the string to set. If the index parameter is -1, the selection is added to or removed from all strings, depending on the value of fSelect.

### Returns

The return value is LB\_ERR if an error occurs.

### Comments

This message should be used only with multiple-selection list boxes.

### See Also

LB\_GETSEL, LB\_SETCURSEL, LB\_SELECTSTRING, LB\_SELITEMRANGE



## LB\_SETTABSTOPS (3.0)

LB\_SETTABSTOPS

```
wParam = (WPARAM) cTabs; /* number of tab stops */  
lParam = (LPARAM) (int FAR*) lpTabs; /* address of tab-stop array */
```

An application sends an LB\_SETTABSTOPS message to set the tab-stop positions in a list box.

Parameter	Description
cTabs	Value of wParam. Specifies the number of tab stops in the list box.
lpTabs	Value of lParam. Points to the first member of an array of integers containing the tab stops, in dialog box units. The tab stops must be sorted in increasing order; back tabs are not allowed.

### Returns

The return value is nonzero if all the tabs were set; otherwise, the return value is zero.

### Comments

To respond to the LB\_SETTABSTOPS message, the list box must have been created with the LBS\_USETABSTOPS style.

If the cTabs parameter is zero and the lpTabs parameter is NULL, the default tab stop is two dialog box units.

If cTabs is 1, the list box will have tab stops separated by the distance specified by lpTabs.

If lpTabs points to more than a single value, a tab stop will be set for each value in lpTabs, up to the number specified by cTabs.

A dialog box unit is a horizontal or vertical distance. One horizontal dialog box unit is equal to one-fourth of the current dialog box base width unit. The dialog box base units are computed based on the height and width of the current system font. The GetDialogBaseUnits function returns the current dialog box base units, in pixels.

## LB\_SETTOPINDEX (3.0)

```
LB_SETTOPINDEX
wParam = (WPARAM) index; /* item index */
lParam = 0L; /* not used, must be zero */
```

An application sends an LB\_SETTOPINDEX message to ensure that a particular item in a list box is visible.

Parameter	Description
index	Value of wParam. Specifies the zero-based index of the item in the list box.

Returns  
The return value is LB\_ERR if an error occurs.

Comments  
The system scrolls the list box so that either the specified item appears at the top of the list box or the maximum scroll range has been reached.

Example  
This example searches for an item in a list box that matches the string "my string" and, if a match is found, ensures that the item is visible:

```
int iIndex;

iIndex = (int) SendMessage(hMyListbox, LB_FINDSTRING, -1,
    (LPARAM) (LPSTR) "my string");

if (iIndex != LB_ERR)
    SendMessage(hMyListbox, LB_SETTOPINDEX, (WPARAM) iIndex, 0);
```

See Also  
LB\_GETTOPINDEX

## STM\_GETICON (3.1)

```
STM_GETICON
wParam = 0; /* not used, must be zero */
lParam = 0L; /* not used, must be zero */
```

An application sends an STM\_GETICON message to retrieve the handle of the icon associated with an icon resource.

### Parameters

This message has no parameters.

### Returns

The return value is the icon handle if the operation is successful, or it is zero if the icon has no associated icon resource or if an error occurred.

### Example

This example gets the handle of the icon associated with an icon resource:

```
HICON hicon;
```

```
hicon = (HICON) SendMessage(hdlg, IDD_ICON,
    STM_GETICON, 0, 0L);
```

### See Also

STM\_SETICON

## STM\_SETICON (3.1)

STM\_SETICON

```
wParam = (WPARAM) (HICON) hicon;    /* handle of the icon*/  
lParam = 0L;    /* not used, must be zero */
```

An application sends an STM\_SETICON message to associate an icon with an icon resource.

Parameter	Description
hicon	Value of wParam. Identifies the icon to associate with the icon resource.

### Returns

The return value is the handle of the icon that was previously associated with the icon resource, or it is zero if an error occurred.

### Example

This example associates the system-defined question-mark icon with an icon resource:

```
HICON hicon, hiconOld;
```

```
hicon = LoadIcon(NULL, IDI_QUESTION);  
hiconOld = (HICON) SendMessage(hdlg, IDD_ICON,  
    STM_SETICON, (WPARAM) hicon, 0);
```

See Also

[STM\\_GETICON](#), [LoadIcon](#)

## WM\_ACTIVATE (2.x)

```
WM_ACTIVATE
fActive = wParam;    /* activation flag */
fMinimized = (BOOL) HIWORD(lParam); /* minimized flag */
hwnd = (HWND) LOWORD(lParam); /* window handle */
```

The WM\_ACTIVATE message is sent when a window is being activated or deactivated. This message is sent first to the window procedure of the main window being deactivated and then to the window procedure of the main window being activated.

Parameter	Description								
fActive	Value of wParam. Specifies whether the window is being activated or deactivated. It can be one of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>WA_INACTIVE</td><td>The window is being deactivated.</td></tr><tr><td>WA_ACTIVE</td><td>The window is being activated through some method other than a mouse click (for example, by use of the keyboard interface to select the window).</td></tr><tr><td>WA_CLICKACTIVE</td><td>The window is being activated by a mouse click.</td></tr></tbody></table>	Value	Description	WA_INACTIVE	The window is being deactivated.	WA_ACTIVE	The window is being activated through some method other than a mouse click (for example, by use of the keyboard interface to select the window).	WA_CLICKACTIVE	The window is being activated by a mouse click.
Value	Description								
WA_INACTIVE	The window is being deactivated.								
WA_ACTIVE	The window is being activated through some method other than a mouse click (for example, by use of the keyboard interface to select the window).								
WA_CLICKACTIVE	The window is being activated by a mouse click.								
fMinimized	Value of the high-order word of lParam. Specifies the minimized state of the window being activated or deactivated. A nonzero value indicates the window is minimized.								
hwnd	Value of the low-order word of lParam. Identifies the window being activated or deactivated. This handle can be NULL.								

### Returns

An application should return zero if it processes this message.

### Comments

If the window is activated with a mouse click, it also receives a WM\_MOUSEACTIVATE message.

### Example

This example sets the input focus while processing the WM\_ACTIVATE message:

```
case WM_ACTIVATE:
    if (wParam && !HIWORD(lParam))
        SetFocus(hwnd);
    break;
```

### See Also

WM\_MOUSEACTIVATE, WM\_NCACTIVATE, DefWindowProc, SetFocus

## WM\_ACTIVATEAPP (2.x)

### WM\_ACTIVATEAPP

```
fActive = (BOOL) wParam; /* the activation/deactivation flag */  
hTask = (HTASK) LOWORD(lParam); /* task handle */
```

The WM\_ACTIVATEAPP message is sent when a window is about to be activated and that window belongs to a different task than the active window. The message is sent to all top-level windows of the task being activated and to all top-level windows of the task being deactivated.

Parameter	Description
fActive	Value of wParam. Specifies whether the window is being activated or deactivated. A nonzero value means the window is being activated. A zero value means the window is being deactivated.
hTask	Value of the low-order word of lParam. Specifies a task handle. If the fActive parameter is nonzero, the handle identifies the task that owns the window being deactivated. If fActive is zero, the handle identifies the task that owns the window being activated.

### Returns

An application should return zero if it processes this message.

### See Also

WM\_ACTIVATE

## WM\_ASKCBFORMATNAME (2.x)

### WM\_ASKCBFORMATNAME

```
wParam = (WPARAM) cbMax; /* maximum bytes to copy */  
lParam = (LPARAM) lpszFormatName; /* address of format name */
```

A clipboard viewer application sends a WM\_ASKCBFORMATNAME message to the clipboard owner when the clipboard contains the data handle of the CF\_OWNERDISPLAY format (that is, when the clipboard owner should display the clipboard contents).

Parameter	Description
cbMax	Value of wParam. Specifies the maximum number of bytes to copy.
lpszFormatName	Value of lParam. Points to the buffer where the copy of the format name is to be stored.

### Returns

An application should return zero if it processes this message.

### Comments

The clipboard owner should copy the name of the CF\_OWNERDISPLAY format into the specified buffer, not exceeding the maximum number of bytes.

### See Also

WM\_PAINTCLIPBOARD

## WM\_CANCELMODE (2.x)

### WM\_CANCELMODE

The WM\_CANCELMODE message is sent to inform a window to cancel any internal mode. This message is sent to the focus window when a dialog box or message box is displayed, giving the focus window the opportunity to cancel modes such as mouse capture.

#### Parameters

This message has no parameters.

#### Returns

An application should return zero if it processes this message.

#### Comments

The DefWindowProc function processes this message by calling the ReleaseCapture function. DefWindowProc does not cancel any other modes.

#### See Also

DefWindowProc, ReleaseCapture



## WM\_CHANGECHAIN (2.x)

### WM\_CHANGECHAIN

```
hwndRemoved = (HWND) wParam; /* handle of removed window */  
hwndNext = (HWND) LOWORD(lParam); /* handle of next window */
```

The WM\_CHANGECHAIN message notifies the first window in the clipboard-viewer chain that a window is being removed from the chain.

Parameter	Description
hwndRemoved	Value of wParam. Identifies the window that is being removed from the clipboard-viewer chain.
hwndNext	Value of the low-order word of lParam. Identifies the window that follows the window being removed from the clipboard-viewer chain.

### Returns

An application should return zero if it processes this message.

### Comments

Each window that receives the WM\_CHANGECHAIN message should call the SendMessage function to pass the message on to the next window in the clipboard-viewer chain. If the window being removed is the next window in the chain, the window specified by the hwndNext parameter becomes the next window and clipboard messages are passed on to it.

### See Also

ChangeClipboardChain, SendMessage

## WM\_CHAR (2.x)

WM\_CHAR

```
nVKey = wParam; /* virtual-key code */  
dwKeyData = (DWORD) lParam; /* key data */
```

The WM\_CHAR message is sent when a WM\_KEYUP message and a WM\_KEYDOWN message are translated. The WM\_CHAR message contains the value of the key being pressed or released.

Parameter	Description																		
nVKey	Value of wParam. Specifies the virtual-key code value of the key.																		
dwKeyData	Value of lParam. Specifies the repeat count, scan code, extended key, context code, previous key state, and key-transition state, as shown in the following table:																		
	<table><thead><tr><th>Bit</th><th>Description</th></tr></thead><tbody><tr><td>0-15</td><td>Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.</td></tr><tr><td>16-23</td><td>Specifies the scan code. The value depends on the original equipment manufacturer (OEM).</td></tr><tr><td>24</td><td>Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.</td></tr><tr><td>25-26</td><td>Not used.</td></tr><tr><td>27-28</td><td>Used internally by Windows.</td></tr><tr><td>29</td><td>Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.</td></tr><tr><td>30</td><td>Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.</td></tr><tr><td>31</td><td>Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.</td></tr></tbody></table>	Bit	Description	0-15	Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.	16-23	Specifies the scan code. The value depends on the original equipment manufacturer (OEM).	24	Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.	25-26	Not used.	27-28	Used internally by Windows.	29	Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.	30	Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.	31	Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.
Bit	Description																		
0-15	Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.																		
16-23	Specifies the scan code. The value depends on the original equipment manufacturer (OEM).																		
24	Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.																		
25-26	Not used.																		
27-28	Used internally by Windows.																		
29	Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.																		
30	Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.																		
31	Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.																		

### Returns

An application should return zero if it processes this message.

### Comments

Because there is not necessarily a one-to-one correspondence between keys pressed and character messages generated, the information in the high-order word of the dwKeyData parameter is usually not useful to applications. The information in the high-order word applies only to the most recent WM\_KEYUP or WM\_KEYDOWN message that precedes the posting of the character message.

For IBM Enhanced 101- and 102-key keyboards, enhanced keys are the right ALT key and the right CTRL key on the main section of the keyboard; the INS, DEL, HOME, END, PAGE UP, PAGE DOWN, and arrow keys in the clusters to the left of the numeric keypad; and the division (/) and ENTER keys on the numeric keypad. Some other keyboards may support the extended-key bit in the dwKeyData parameter.

### See Also

WM\_KEYDOWN, WM\_KEYUP

## WM\_CHAROITEM (3.0)

```
WM_CHAROITEM
nKey = wParam; /* key value */
hwndListBox = (HWND) LOWORD(lParam); /* list box handle */
iCaretPos = HIWORD(lParam); /* caret position */
```

The WM\_CHAROITEM message is sent by a list box with the LBS\_WANTKEYBOARDINPUT style to its owner in response to a WM\_CHAR message.

Parameter	Description
nKey	Value of wParam. Specifies the value of the key the user pressed.
hwndListBox	Value of the low-order word of lParam. Identifies the list box.
iCaretPos	Value of the high-order word of lParam. Specifies the current caret position.

### Returns

The return value specifies the action that the application performed in response to the message. A return value of -2 indicates that the application handled all aspects of selecting the item and requires no further action by the list box. A return value of -1 indicates that the list box should perform the default action in response to the keystroke. A return value of 0 or greater specifies the zero-based index of an item in the list box and indicates that the list box should perform the default action for the keystroke on the given item.

### Comments

Only owner-drawn list boxes that do not have the LBS\_HASSTRINGS style can receive this message.

### See Also

WM\_CHAR, WM\_VKEYTOITEM

## WM\_CHILDACTIVATE (2.x)

### WM\_CHILDACTIVATE

The WM\_CHILDACTIVATE message is sent to a multiple document interface (MDI) child window when the user clicks the window's title bar or when the window is activated, moved, or sized.

#### Parameters

This message has no parameters.

#### Returns

An application should return zero if it processes this message.

#### See Also

[MoveWindow](#), [SetWindowPos](#)

## WM\_CHOOSEFONT\_GETLOGFONT (3.1)

```
WM_CHOOSEFONT_GETLOGFONT
wParam = 0;      /* not used, must be zero */
lpLpFont = (LPLOGFONT) lParam; /* address of a LOGFONT structure */
```

An application sends a WM\_CHOOSEFONT\_GETLOGFONT message to the Font dialog box created by the ChooseFont function to retrieve the current LOGFONT structure.

Parameter	Description
lpLpFont	Points to a <u>LOGFONT</u> structure that receives information about the current logical font.

### Returns

This message does not return a value.

### Comments

An application uses this message to retrieve the LOGFONT structure while the Font dialog box is open. When the user closes the dialog box, the ChooseFont function receives information about the LOGFONT structure.

### See Also

WM\_GETFONT, ChooseFont, LOGFONT

## WM\_CLEAR (2.x)

```
WM_CLEAR
wParam = 0; /* not used, must be zero */
lParam = 0L; /* not used, must be zero */
```

An application sends a WM\_CLEAR message to an edit control or combo box to delete (clear) the current selection, if any, in the edit control.

### Parameters

This message has no parameters.

### Returns

The return value is nonzero if this message is sent to an edit control or a combo box.

### Comments

The deletion performed by the WM\_CLEAR message can be undone by sending the edit control an EM\_UNDO message.

To delete the current selection and place the deleted contents into the clipboard, use the WM\_CUT message.

### Example

This example sends an EM\_SETSEL message to select the entire contents of an edit control. It then sends a WM\_CLEAR message to delete the contents of the edit control.

```
SendMessage(hdlg, ID_MYEDITCONTROL,  
            EM_SETSEL, 0, MAKELONG(0, -1));  
SendMessage(hdlg, ID_MYEDITCONTROL,  
            WM_CLEAR, 0, 0);
```

### See Also

EM\_UNDO, WM\_COPY, WM\_CUT, WM\_PASTE

## WM\_CLOSE (2.x)

```
WM_CLOSE
wParam = 0; /* not used, must be zero */
lParam = 0L; /* not used, must be zero */
```

The WM\_CLOSE message is sent as a signal that a window or an application should terminate. An application can prompt the user for confirmation prior to destroying the window by processing the WM\_CLOSE message and calling the DestroyWindow function only if the user confirms the choice.

### Parameters

This message has no parameters.

### Returns

An application should return zero if it processes this message.

### Example

This example processes a WM\_CLOSE message and requests confirmation from the user before terminating the application:

```
case WM_CLOSE:
    if (MessageBox(hwnd, "Are you sure you want to exit?", "MyApp",
        MB_ICONQUESTION | MB_OKCANCEL) == IDOK)
        DestroyWindow(hwnd);
    return 0L;
```

### See Also

DestroyWindow, PostQuitMessage, WM\_DESTROY, WM\_QUERYENDSESSION, WM\_QUIT

## WM\_COMMAND (2.x)

```
WM_COMMAND
idItem = wParam; /* control or menu item identifier */
hwndCtl = (HWND) LOWORD(lParam); /* handle of control */
wNotifyCode = HIWORD(lParam); /* notification message */
```

The WM\_COMMAND message is sent to a window when the user selects an item from a menu, when a control sends a notification message to its parent window, or when an accelerator keystroke is translated.

Parameter	Description
idItem	Value of wParam. Specifies the identifier of the menu item or control.
hwndCtl	Value of the low-order word of lParam. Identifies the control sending the message if the message is from a control. Otherwise, this parameter is zero.
wNotifyCode	Value of the high-order word of lParam. Specifies the notification message if the message is from a control. If the message is from an accelerator, this parameter is 1. If the message is from a menu, this parameter is 0.

### Returns

An application should return zero if it processes this message.

### Comments

Accelerator keystrokes that are defined to select items from the System menu (sometimes referred to as the Control menu) are translated into WM\_SYSCOMMAND messages.

If an accelerator keystroke that corresponds to a menu item occurs when the window that owns the menu is minimized, no WM\_COMMAND message is sent. However, if an accelerator keystroke occurs that does not match any of the items on the window's menu or on the System menu, a WM\_COMMAND message is sent even if the window is minimized.

### Example

This example creates an Options dialog box in response to a WM\_COMMAND message sent as a result of a menu selection:

```
FARPROC lpProc;

case WM_COMMAND:
    switch (wParam) {
        case IDM_OPTIONS:
            lpProc = MakeProcInstance(OptionsProc, hInstance);
            DialogBox(hInstance, "OptionsBox", hwnd, (DLGPROC) lpProc);
            FreeProcInstance(lpProc);
            break;

        . /* Process other menu commands. */
        .
    }
    break;
```

### See Also

WM\_SYSCOMMAND



## WM\_COMMNOTIFY (3.1)

```
WM_COMMNOTIFY
idDevice = wParam; /* communication-device ID */
nNotifyStatus = LOWORD(lParam); /* notification-status flag */
```

The WM\_COMMNOTIFY message is posted by a communication device driver whenever a COM port event occurs. The message indicates the status of a window's input or output queue.

Parameter	Description								
idDevice	Value of wParam. Specifies the identifier of the communication device that is posting the notification message.								
nNotifyStatus	Value of the low-order word of lParam. Specifies the notification status in the low-order word. The notification status may be one or more of the following flags: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>CN_EVENT</td><td>Indicates that an event has occurred that was enabled in the event word of the communication device. This event was enabled by a call to the <u>SetCommEventMask</u> function. The application should call the <u>GetCommEventMask</u> function to determine which event occurred and to clear the event.</td></tr><tr><td>CN_RECEIVE</td><td>Indicates that at least cbWriteNotify bytes are in the input queue. The cbWriteNotify parameter is a parameter of the <u>EnableCommNotification</u> function.</td></tr><tr><td>CN_TRANSMIT</td><td>Indicates that fewer than cbOutQueue bytes are in the output queue waiting to be transmitted. The cbOutQueue parameter is a parameter of the <u>EnableCommNotification</u> function.</td></tr></tbody></table>	Value	Meaning	CN_EVENT	Indicates that an event has occurred that was enabled in the event word of the communication device. This event was enabled by a call to the <u>SetCommEventMask</u> function. The application should call the <u>GetCommEventMask</u> function to determine which event occurred and to clear the event.	CN_RECEIVE	Indicates that at least cbWriteNotify bytes are in the input queue. The cbWriteNotify parameter is a parameter of the <u>EnableCommNotification</u> function.	CN_TRANSMIT	Indicates that fewer than cbOutQueue bytes are in the output queue waiting to be transmitted. The cbOutQueue parameter is a parameter of the <u>EnableCommNotification</u> function.
Value	Meaning								
CN_EVENT	Indicates that an event has occurred that was enabled in the event word of the communication device. This event was enabled by a call to the <u>SetCommEventMask</u> function. The application should call the <u>GetCommEventMask</u> function to determine which event occurred and to clear the event.								
CN_RECEIVE	Indicates that at least cbWriteNotify bytes are in the input queue. The cbWriteNotify parameter is a parameter of the <u>EnableCommNotification</u> function.								
CN_TRANSMIT	Indicates that fewer than cbOutQueue bytes are in the output queue waiting to be transmitted. The cbOutQueue parameter is a parameter of the <u>EnableCommNotification</u> function.								

### Returns

An application should return zero if it processes this message.

### Comments

This message is sent only when the event word changes for the communication device. The application that sends WM\_COMMNOTIFY must clear each event to be sure of receiving future notifications.

### See Also

[EnableCommNotification](#)

## WM\_COMPACTING (3.0)

WM\_COMPACTING

```
wCompactRatio = wParam; /* compacting ratio */
```

The WM\_COMPACTING message is sent to all top-level windows when Windows detects that more than 12.5 percent of system time over a 30- to 60-second interval is being spent compacting memory. This indicates that system memory is low.

Parameter	Description
wCompactRatio	Value of wParam. Specifies the ratio of central processing unit (CPU) time currently spent by Windows compacting memory to CPU time currently spent by Windows performing other operations. For example, 0x8000 represents 50 percent of CPU time spent compacting memory.

### Returns

An application should return zero if it processes this message.

### Comments

When an application receives this message, it should free as much memory as possible, taking into account the current level of activity of the application and the total number of applications running with Windows. The application can call the [GetNumTasks](#) function to determine how many applications are running.

### See Also

[GetNumTasks](#)

## WM\_COMPAREITEM (3.0)

```
WM_COMPAREITEM
idCtl = wParam; /* control identifier */
lpcis = (const COMPAREITEMSTRUCT FAR*) lParam; /* structure*/
```

The WM\_COMPAREITEM message determines the relative position of a new item in the sorted list of an owner-drawn combo box or list box. Whenever the application adds a new item, Windows sends this message to the owner of a combo box or list box created with the CBS\_SORT or LBS\_SORT style.

Parameter	Description
idCtl	Value of wParam. Specifies the identifier of the control that sent the WM_COMPAREITEM message.
lpcis	Value of lParam. Points to a <u>COMPAREITEMSTRUCT</u> data structure that contains the identifiers and application-supplied data for two items in the combo box or list box.

### Returns

The return value indicates the relative position of the two items. It may be any of the following values:

Value	Meaning
-1	Item 1 precedes item 2 in the sorted order.
0	Item 1 and item 2 are equivalent in the sorted order.
1	Item 1 follows item 2 in the sorted order.

### Comments

When the owner of an owner-drawn combo box or list box receives this message, the owner returns a value indicating which of the items specified in the COMPAREITEMSTRUCT structure should appear before the other. Typically, Windows sends this message several times until it determines the exact position for the new item.

### See Also

[COMPAREITEMSTRUCT](#)

### Windows 3.1 changes

The meaning of the `wParam` parameter has changed. The `wParam` parameter specifies the identifier of the control.

## WM\_COPY (2.x)

```
WM_COPY
wParam = 0; /* not used, must be zero */
lParam = 0L; /* not used, must be zero */
```

An application sends a WM\_COPY message to an edit control or combo box to copy the current selection to the clipboard in CF\_TEXT format.

### Parameters

This message has no parameters.

### Returns

The return value is nonzero if this message is sent to an edit control or a combo box.

### Example

This example sends an EM\_SETSEL message to select the entire contents of an edit control. It then sends a WM\_COPY message to copy the contents of the edit control to the clipboard.

```
SendDlgItemMessage(hdlg, ID_MYEDITCONTROL,  
EM_SETSEL, 0, MAKELONG(0, -1));  
SendDlgItemMessage(hdlg, ID_MYEDITCONTROL,  
WM_COPY, 0, 0);
```

### See Also

WM\_CLEAR, WM\_CUT, WM\_PASTE

## WM\_CREATE (2.x)

WM\_CREATE

```
lpcs = (CREATESTRUCT FAR*) lParam; /* structure address*/
```

The WM\_CREATE message is sent when an application requests that a window be created by calling the CreateWindowEx or CreateWindow function. The window procedure for the new window receives this message after the window is created but before the window becomes visible. The message is sent to the window before the CreateWindowEx or CreateWindow function returns.

Parameter	Description
lpcs	Value of lParam. Points to a <u>CREATESTRUCT</u> data structure containing information about the window being created. The members of the <u>CREATESTRUCT</u> structure are identical to the parameters of the <u>CreateWindowEx</u> function.

### Returns

If an application processes this message, it should return 0 to continue creation of the window. If the application returns -1, the window will be destroyed and the CreateWindowEx or CreateWindow function will return a NULL handle.

### See Also

CreateWindow, CreateWindowEx, WM\_NCCREATE, CREATESTRUCT

## WM\_CTLCOLOR (2.x)

```
WM_CTLCOLOR
hdcChild = (HDC) wParam; /* child-window display context */
hwndChild = (HWND) LOWORD(lParam); /* handle of child window */
nCtlType = (int) HIWORD(lParam); /* type of control */
```

The WM\_CTLCOLOR message is sent to the parent of a system-defined control class or a message box when the control or message box is about to be drawn. The following controls send this message:

- Combo boxes
- Edit controls
- List boxes
- Buttons
- Static controls
- Scroll bars

Parameter	Description
hdcChild	Value of wParam. Identifies the display context for the child window.
hwndChild	Value of the low-order word of lParam. Identifies the child window.
nCtlType	Value of the high-order word of lParam. Specifies the type of the control. This parameter can be one of the following values:

Value	Meaning
CTLCOLOR_BTN	Button
CTLCOLOR_DLG	Dialog box
CTLCOLOR_EDIT	Edit control
CTLCOLOR_LISTBOX	List box
CTLCOLOR_MSGBOX	Message box
CTLCOLOR_SCROLLBAR	Scroll bar
CTLCOLOR_STATIC	Static control

### Returns

If an application processes the WM\_CTLCOLOR message, it must return a handle to the brush that is to be used for painting the control background or it must return NULL.

### Comments

The WM\_CTLCOLOR message is sent to the parent window for all control types except dialog boxes. When the nCtlType parameter specifies CTLCOLOR\_DLG, the message is sent to the dialog box procedure.

To change the text color, the application should call the [SetTextColor](#) function with the desired red, green, and blue (RGB) values.

To change the background color of a single-line edit control, the application must set the brush handle in both the CTLCOLOR\_EDIT and CTLCOLOR\_MSGBOX message codes, and the application must call the [SetBkColor](#) function in response to the CTLCOLOR\_EDIT code.

The return value from this message has no effect on a button with the [BS\\_PUSHBUTTON](#) or [BS\\_DEFPUSHBUTTON](#) style.

To change the color of the list box for a drop-down combo box, applications should subclass the combo box and check for the WM\_CTLCOLOR message with CTLCOLOR\_LISTBOX in the nCtlType parameter. This procedure can return a handle to the brush that will be used to paint the background. In this case, the [SetBkColor](#) function must be used to set the background color for the text.

### Example

This example creates a green brush and passes the handle of the brush to a single-line edit control in response to a WM\_CTLCOLOR message:

```
static HBRUSH hbrGreen;

switch(msg) {
    case WM_INITDIALOG:

        /* Create a green brush */
```

```

    hbrGreen = CreateSolidBrush(RGB(0, 255, 0));
    return TRUE;

    case WM_CTLCOLOR:
        switch(HIWORD(lParam)) {
        case CTLCOLOR_EDIT:

/* Set text to white and background to green */

SetTextColor((HDC) wParam, RGB(255, 255, 255));
SetBkColor((HDC) wParam, RGB(0, 255, 0));
return (LRESULT) hbrGreen;

        case CTLCOLOR_MSGBOX:

/*
 * For single-line edit controls, this code must be
 * processed so that the background color of the format
 * rectangle will also be painted with the new color.
 */

return (LRESULT) hbrGreen;
        }
        return (LRESULT) NULL;
}

See Also
SetBkColor

```



## WM\_CUT (2.x)

WM\_CUT

```
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends a WM\_CUT message to an edit control or combo box to delete (cut) the current selection, if any, in the edit control and copy the deleted text to the clipboard in CF\_TEXT format.

Parameters

This message has no parameters.

Returns

The return value is nonzero if this message is sent to an edit control or a combo box.

Comments

An EM\_UNDO message can be sent to the edit control to undo the deletion performed by the WM\_CUT message.

To delete the current selection without placing the deleted text onto the clipboard, use the WM\_CLEAR message.

Example

This example sends an EM\_SETSEL message to select the entire contents of an edit control. It then sends a WM\_CUT message to delete the contents of the edit control and to copy the deleted text to the clipboard.

```
SendMessage(hdlg, ID_MYEDITCONTROL,  
            EM_SETSEL, 0, MAKELONG(0, -1));  
SendMessage(hdlg, ID_MYEDITCONTROL,  
            WM_CUT, 0, 0);
```

See Also

WM\_CLEAR, WM\_COPY, WM\_PASTE

## WM\_DDE\_ACK (2.x)

```
#include <dde.h>
```

```
WM_DDE_ACK
```

```
wParam = (WPARAM) hwnd; /* handle of posting window */
```

```
lParam = MAKELPARAM(wLow, wHigh); /* depending on received message */
```

The WM\_DDE\_ACK message notifies an application of the receipt and processing of a WM\_DDE\_INITIATE, WM\_DDE\_EXECUTE, WM\_DDE\_DATA, WM\_DDE\_ADVISE, WM\_DDE\_UNADVISE, or WM\_DDE\_POKE message, and in some cases, of a WM\_DDE\_REQUEST message.

Parameter

Description

hwnd

Value of wParam. Specifies the handle of the window posting the message.

wLow

Value of the low-order word of lParam. Specifies data as follows, depending on the message to which the WM\_DDE\_ACK message is responding:

Message

Parameter

Description

WM\_DDE\_INITIATE

aApplication

An atom that contains the name of the replying application.

WM\_DDE\_EXECUTE and all other messages

wStatus

A series of flags that indicate the status of the response.

wHigh

Value of high-order word of lParam. Specifies data as follows, depending on the message to which the WM\_DDE\_ACK message is responding:

Message

Parameter

Description

WM\_DDE\_INITIATE

aTopic

An atom that contains the topic with which the replying server window is associated.

WM\_DDE\_EXECUTE

hCommands

A handle that identifies the data item containing the command string.

All other messages

aItem

An atom that specifies the data item for which the response is sent.

Returns

This message does not return a value.

Comments

The wStatus word consists of a DDEACK data structure.

Posting

Except in response to the WM\_DDE\_INITIATE message, the application posts the WM\_DDE\_ACK message by calling the PostMessage function, not the SendMessage function. When responding to WM\_DDE\_INITIATE, the application sends the WM\_DDE\_ACK message by calling SendMessage.

When acknowledging any message with an accompanying aItem atom, the application posting WM\_DDE\_ACK can either reuse the aItem atom that accompanied the original message or delete it and create a new one.

When acknowledging WM\_DDE\_EXECUTE, the application that posts WM\_DDE\_ACK should reuse the hCommands object that accompanied the original WM\_DDE\_EXECUTE message.

If an application has initiated the termination of a conversation by posting WM\_DDE\_TERMINATE and is awaiting confirmation, the waiting application should not acknowledge (positively or negatively) any subsequent messages sent by the other application. The waiting application should delete any atoms or shared memory objects received in these intervening messages (but should not delete the atoms in response to the WM\_DDE\_ACK message).

Receiving

The application that receives WM\_DDE\_ACK should delete all atoms accompanying the message.

If the application receives WM\_DDE\_ACK in response to a message with an accompanying hData object, the application should delete the hData object.

If the application receives a negative WM\_DDE\_ACK message posted in reply to a WM\_DDE\_ADVISE message, the application should delete the hOptions object posted with the original WM\_DDE\_ADVISE message.

If the application receives a negative WM\_DDE\_ACK message posted in reply to a WM\_DDE\_EXECUTE message, the application should delete the hCommands object posted with the original WM\_DDE\_EXECUTE message.

See Also

DDEACK, PostMessage, WM\_DDE\_ADVISE, WM\_DDE\_DATA, WM\_DDE\_EXECUTE, WM\_DDE\_INITIATE, WM\_DDE\_POKE, WM\_DDE\_REQUEST, WM\_DDE\_TERMINATE, WM\_DDE\_UNADVISE, DDEACK

## WM\_DDE\_ADVISE (2.x)

```
#include <dde.h>
```

```
WM_DDE_ADVISE
```

```
wParam = (WPARAM) hwnd; /* handle of posting window */  
lParam = MAKELPARAM(hOptions, aItem); /* send options and data item */
```

A dynamic data exchange (DDE) client application posts the WM\_DDE\_ADVISE message to a DDE server application to request the server to supply an update for a data item whenever it changes.

Parameter	Description
hwnd	Value of wParam. Identifies the sending window.
hOptions	Value of the low-order word of lParam. Specifies a handle of a global memory object that specifies how the data is to be sent.
aItem	Value of the high-order word of lParam. Specifies the data item being requested.

### Returns

This message does not return a value.

### Comments

The global memory object identified by the hOptions parameter consists of a DDEADVISE data structure.

If an application supports more than one clipboard format for a single topic and item, it can post multiple WM\_DDE\_ADVISE messages for the topic and item, specifying a different clipboard format with each message.

### Posting

The application posts the WM\_DDE\_ADVISE message by calling the PostMessage function, not the SendMessage function.

The application allocates hOptions by calling the GlobalAlloc function with the GMEM\_DDESHARE option.

The application allocates aItem by calling the GlobalAddAtom function.

If the receiving (server) application responds with a negative WM\_DDE\_ACK message, the posting (client) application must delete the hOptions object.

### Receiving

The application posts the WM\_DDE\_ACK message to respond positively or negatively. When posting WM\_DDE\_ACK, the application can reuse the aItem atom or delete it and create a new one. If the WM\_DDE\_ACK message is positive, the application should delete the hOptions object; otherwise, the application should not delete the object.

### See Also

DDEADVISE, GlobalAddAtom, GlobalAlloc, PostMessage, WM\_DDE\_DATA, WM\_DDE\_REQUEST, DDEADVISE

## WM\_DDE\_DATA (2.x)

```
#include <dde.h>
```

```
WM_DDE_DATA
```

```
wParam = (WPARAM) hwnd; /* handle of posting window */  
lParam = MAKELPARAM(hData, aItem); /* memory object and data item */
```

A dynamic data exchange (DDE) server application posts a WM\_DDE\_DATA message to a DDE client application to pass a data item to the client or to notify the client of the availability of a data item.

Parameter	Description
hwnd	Value of wParam. Specifies the handle of the window posting the message.
hData	Value of the low-order word of lParam. Identifies the global memory object containing the data and additional information. The handle should be set to NULL if the server is notifying the client that the data item value has changed during a warm link. A warm link is established when the client sends a <u>WM_DDE_ADVISE</u> message with the fDeferUpd bit set.
aItem	Value of the high-order word of lParam. Specifies the data item for which data or notification is sent.

### Returns

This message does not return a value.

### Comments

The global memory object identified by the hData parameter consists of a DDEDATA structure.

### Posting

The application posts the WM\_DDE\_DATA message by calling the PostMessage function, not the SendMessage function.

The application allocates hData by calling the GlobalAlloc function with the GMEM\_DDESHARE option.

The application allocates aItem by calling the GlobalAddAtom function.

If the receiving (client) application responds with a negative WM\_DDE\_ACK message, the posting (server) application must delete the hData object.

If the posting (server) application sets the fRelease member of the DDEDATA structure to FALSE, the posting application is responsible for deleting hData upon receipt of either a positive or negative acknowledgment.

The application should not set both the fAckReq and fRelease members of the DDEDATA structure to FALSE. If both members are set to FALSE, it is difficult for the posting (server) application to determine when to delete hData.

### Receiving

If fAckReq is TRUE, the application posts the WM\_DDE\_ACK message to respond positively or negatively. When posting WM\_DDE\_ACK, the application can reuse the aItem atom or delete it and create a new one.

If fAckReq is FALSE, the application deletes the aItem atom.

If the posting (server) application specified hData as NULL, the receiving (client) application can request the server to send the actual data by posting a WM\_DDE\_REQUEST message.

After processing a WM\_DDE\_DATA message in which hData is not NULL, the application should delete hData unless either of the following conditions is true:

- The fRelease member is FALSE.
- The fRelease member is TRUE, but the receiving (client) application responds with a negative WM\_DDE\_ACK message.

### See Also

DDEDATA, GlobalAddAtom, GlobalAlloc, PostMessage, WM\_DDE\_ACK, WM\_DDE\_ADVISE, WM\_DDE\_POKE, WM\_DDE\_REQUEST

## WM\_DDE\_EXECUTE (2.x)

```
#include <dde.h>
```

```
WM_DDE_EXECUTE
```

```
wParam = (WPARAM) hwnd; /* handle of posting window */  
lParam = MAKELPARAM(reserved, hCommands); /* commands to execute */
```

A dynamic data exchange (DDE) client application posts a WM\_DDE\_EXECUTE message to a DDE server application to send a string to the server to be processed as a series of commands. The server application is expected to post a WM\_DDE\_ACK message in response.

Parameter	Description
hwnd	Value of wParam. Identifies the sending window.
reserved	Value of the low-order word of lParam. Reserved; must be zero.
hCommands	Value of the high-order word of lParam. Identifies a global memory object containing the command(s) to be executed.

### Returns

This message does not return a value.

### Comments

The command string is a null-terminated string, consisting of one or more opcode strings enclosed in single brackets ([ ]) and separated by spaces.

Each opcode string has the following syntax. The parameters list is optional.

#### opcode parameters

The opcode is any application-defined single token. It cannot include spaces, commas, parentheses, or quotation marks.

The parameters list can contain any application-defined value or values. Multiple parameters are separated by commas, and the entire parameter list is enclosed in parentheses. Parameters cannot include commas or parentheses except inside a quoted string. If a bracket or parenthesis character is to appear in a quoted string, it must be doubled--for example, "(".

The following are valid command strings:

```
[connect][download(query1,results.txt)][disconnect]  
[query("sales per employee for each district")]  
[open("sample.xml")][run("r1c1")]
```

### Posting

The application posts the WM\_DDE\_EXECUTE message by calling the PostMessage function, not the SendMessage function.

The application allocates hCommands by calling the GlobalAlloc function with the GMEM\_DDESHARE option.

When processing a WM\_DDE\_ACK message posted in reply to a WM\_DDE\_EXECUTE message, the application that posted the original WM\_DDE\_EXECUTE message must delete the hCommands object sent back in the WM\_DDE\_ACK message.

### Receiving

The application posts the WM\_DDE\_ACK message to respond positively or negatively, reusing the hCommands object.

### See Also

PostMessage, WM\_DDE\_ACK

## WM\_DDE\_INITIATE (2.x)

```
#include <dde.h>
```

```
WM_DDE_INITIATE  
wParam = (WPARAM) hwnd; /* sending window's handle */  
lParam = MAKELPARAM(aApplication, aTopic); /* app. and topic atoms */  
/
```

A dynamic data exchange (DDE) client application sends a WM\_DDE\_INITIATE message to initiate a conversation with server applications responding to the specified application and topic names.

Upon receiving this message, all server applications with names that match the aApplication application and that support the aTopic topic are expected to acknowledge it (see the WM\_DDE\_ACK message).

Parameter	Description
hwnd	Value of wParam. Identifies the sending window.
aApplication	Value of the low-order word of lParam. Specifies the atom identifying the name of the application with which a conversation is requested. The application name cannot contain slash marks (/) or backslashes (\). These characters are reserved for future use in network implementations. If aApplication is NULL, a conversation with all applications is requested.
aTopic	Value of the high-order word of lParam. Specifies the atom identifying the the topic for which a conversation is requested. If the topic is NULL, a conversation for all available topics is requested.

### Returns

This message does not return a value.

### Comments

If aApplication is NULL, any application can respond. If aTopic is NULL, any topic is valid. Upon receiving a WM\_DDE\_INITIATE request with the aTopic parameter set to NULL, an application is expected to send a WM\_DDE\_ACK message for each of the topics it supports.

### Sending

The application sends the WM\_DDE\_INITIATE message by calling the SendMessage function, not the PostMessage function. The application broadcasts the message to all windows by setting the first parameter of SendMessage to -1, as shown:

```
SendMessage(-1, WM_DDE_INITIATE, hwndClient, MAKELONG(aApp, aTopic));
```

If the application has already obtained the window handle of the desired server, it can send WM\_DDE\_INITIATE directly to the server window by passing the server's window handle as the first parameter of SendMessage.

The application allocates aApplication and aTopic by calling GlobalAddAtom.

When SendMessage returns, the application deletes the aApplication and aTopic atoms.

### Receiving

To complete the initiation of a conversation, the application responds with one or more WM\_DDE\_ACK messages, where each message is for a separate topic. When sending a WM\_DDE\_ACK message, the application creates new aApplication and aTopic atoms; it should not reuse the atoms sent with the WM\_DDE\_INITIATE message.

### See Also

GlobalAddAtom, SendMessage, WM\_DDE\_ACK

## WM\_DDE\_POKE (2.x)

```
#include <dde.h>
```

```
WM_DDE_POKE
```

```
wParam = (WPARAM) hwnd; /* handle of posting window */  
lParam = MAKELPARAM(hData, aItem); /* data handle and item*/
```

A dynamic data exchange (DDE) client application posts a WM\_DDE\_POKE message to a server application. A client uses this message to request the server to accept an unsolicited data item. The server is expected to reply with a WM\_DDE\_ACK message indicating whether it accepted the data item.

Parameter	Description
hwnd	Value of wParam. Specifies the handle of the window posting the message.
hData	Value of the low-order word of lParam. Identifies the data being posted. The handle identifies a global memory object that contains a <u>DDEPOKE</u> data structure.
aItem	Value of the high-order word of lParam. Specifies a global atom that identifies the data item being offered to the server.

### Returns

This message does not return a value.

### Comments

#### Posting

The posting (client) application should do the following:

- Use the PostMessage function to post the WM\_DDE\_POKE message.
- Use the GlobalAlloc function with the GMEM\_DDESHARE option to allocate memory for the data.
- Use the GlobalAddAtom function to create the atom for the data item.
- Delete the global memory object if the server application responds with a negative WM\_DDE\_ACK message.
- Delete the global memory object if the client has set the fRelease member of the DDEPOKE structure to FALSE and the server responds with either a positive or negative WM\_DDE\_ACK.

#### Receiving

The receiving (server) application should do the following:

- Post the WM\_DDE\_ACK message to respond positively or negatively. When posting WM\_DDE\_ACK, reuse the data-item atom or delete it and create a new one.
- Delete the global memory object after processing WM\_DDE\_POKE unless either the fRelease flag was set to FALSE or the fRelease flag was set to TRUE but the server has responded with a negative WM\_DDE\_ACK message.

#### See Also

DDEPOKE, GlobalAlloc, PostMessage, WM\_DDE\_ACK, WM\_DDE\_DATA, DDEPOKE



## WM\_DDE\_REQUEST (2.x)

```
#include <dde.h>
```

```
WM_DDE_REQUEST
```

```
wParam = (WPARAM) hwnd; /* handle of posting window */  
lParam = MAKELPARAM(cfFormat, aItem); /* clipboard format and item */
```

A dynamic data exchange (DDE) client application posts a WM\_DDE\_REQUEST message to a DDE server application to request the value of a data item.

Parameter	Description
hwnd	Value of wParam. Identifies the sending window.
cfFormat	Value of the low-order word of lParam. Specifies a standard or registered clipboard format number.
aItem	Value of the high-order word of lParam. Specifies which data item is being requested from the server.

### Returns

This message does not return a value.

### Comments

#### Posting

The application posts the WM\_DDE\_REQUEST message by calling the PostMessage function, not the SendMessage function.

The application allocates aItem by calling the GlobalAddAtom function.

#### Receiving

If the receiving (server) application can satisfy the request, it responds with a WM\_DDE\_DATA message containing the requested data. Otherwise, it responds with a negative WM\_DDE\_ACK message.

When responding with either a WM\_DDE\_DATA or WM\_DDE\_ACK message, the application can reuse the aItem atom or delete it and create a new one.

### See Also

GlobalAddAtom, PostMessage, WM\_DDE\_ACK

## WM\_DDE\_TERMINATE (2.x)

```
#include <dde.h>
```

```
WM_DDE_TERMINATE
```

```
wParam = (WPARAM) hwnd; /* handle of posting window */  
lParam = 0L; /* not used, must be zero */
```

A dynamic data exchange (DDE) application (client or server) posts a WM\_DDE\_TERMINATE message to terminate a conversation.

Parameter	Description
hwnd	Value of wParam. Identifies the sending window.

### Returns

This message does not return a value.

### Comments

#### Posting

The application posts the WM\_DDE\_TERMINATE message by calling the PostMessage function, not the SendMessage function.

While waiting for confirmation of the termination, the posting application should not acknowledge any other messages sent by the receiving application. If the posting application receives messages (other than WM\_DDE\_TERMINATE) from the receiving application, it should delete any atoms or shared memory objects accompanying the messages.

#### Receiving

The application responds by posting a WM\_DDE\_TERMINATE message.

### See Also

PostMessage

## WM\_DDE\_UNADVISE (2.x)

```
#include <dde.h>
```

```
WM_DDE_UNADVISE  
wParam = (WPARAM) hwnd; /* handle of posting window */  
lParam = MAKELPARAM(cfFormat, aItem); /* clipboard format and item */  
/
```

A dynamic data exchange (DDE) client application posts a WM\_DDE\_UNADVISE message to inform a server application that the specified item or a particular clipboard format for the item should no longer be updated. This terminates the warm or hot link for the specified item.

Parameter	Description
hwnd	Value of wParam. Identifies the sending window.
cfFormat	Value of the low-order word of lParam. Specifies the clipboard format of the item for which the update request is being retracted. When the cfFormat parameter is NULL, all active <u>WM_DDE_ADVISE</u> conversations for the item are to be terminated.
aItem	Value of the high-order word of lParam. Specifies the item for which the update request is being retracted. When aItem is NULL, all active <u>WM_DDE_ADVISE</u> conversations associated with the client are to be terminated.

### Returns

This message does not return a value.

### Comments

#### Posting

The application posts the WM\_DDE\_UNADVISE message by calling the PostMessage function, not the SendMessage function.

The application allocates aItem by calling the GlobalAddAtom function.

#### Receiving

The application posts the WM\_DDE\_ACK message to respond positively or negatively. When posting WM\_DDE\_ACK, the application can reuse the aItem atom or delete it and create a new one.

### See Also

GlobalAddAtom, PostMessage, WM\_DDE\_ACK

## WM\_DEADCHAR (2.x)

### WM\_DEADCHAR

```
chDeadKey = wParam; /* dead-key character */
dwKeyData = (DWORD) lParam; /* key data */
```

The WM\_DEADCHAR message is sent when a WM\_KEYUP message and a WM\_KEYDOWN message are translated. It specifies the character value of a dead key. A dead key is a key, such as the umlaut (double-dot) character, that is combined with other characters to form a composite character. For example, the umlaut-O character consists of the dead key, umlaut, and the O key.

Parameter	Description																		
chDeadKey	Value of wParam. Specifies the dead-key character value.																		
dwKeyData	Value of lParam. Specifies the repeat count, scan code, extended key, context code, previous key state, and key-transition state, as shown in the following table:																		
	<table><thead><tr><th>Bit</th><th>Description</th></tr></thead><tbody><tr><td>0-15</td><td>Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.</td></tr><tr><td>16-23</td><td>Specifies the scan code. The value depends on the original equipment manufacturer (OEM).</td></tr><tr><td>24</td><td>Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.</td></tr><tr><td>25-26</td><td>Not used.</td></tr><tr><td>27-28</td><td>Used internally by Windows.</td></tr><tr><td>29</td><td>Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.</td></tr><tr><td>30</td><td>Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.</td></tr><tr><td>31</td><td>Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.</td></tr></tbody></table>	Bit	Description	0-15	Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.	16-23	Specifies the scan code. The value depends on the original equipment manufacturer (OEM).	24	Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.	25-26	Not used.	27-28	Used internally by Windows.	29	Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.	30	Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.	31	Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.
Bit	Description																		
0-15	Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.																		
16-23	Specifies the scan code. The value depends on the original equipment manufacturer (OEM).																		
24	Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.																		
25-26	Not used.																		
27-28	Used internally by Windows.																		
29	Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.																		
30	Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.																		
31	Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.																		

### Returns

An application should return zero if it processes this message.

### Comments

An application typically uses the WM\_DEADCHAR message to give the user feedback about each key pressed. For example, an application can display the accent in the current character position without moving the caret.

Because there is not necessarily a one-to-one correspondence between keys pressed and character messages generated, the information in the high-order word of the dwKeyData parameter is usually not useful to applications. The information in the high-order word applies only to the most recent WM\_KEYUP or WM\_KEYDOWN message that precedes the posting of the character message.

For IBM Enhanced 101- and 102-key keyboards, enhanced keys are the right ALT key and the right CTRL key on the main section of the keyboard; the INS, DEL, HOME, END, PAGE UP, PAGE DOWN, and arrow keys in the clusters to the left of the numeric keypad; and the division (/) and ENTER keys on the numeric keypad. Some other keyboards may support the extended-key bit in the dwKeyData parameter.

### See Also

WM\_KEYDOWN

■ WM\_DELETEITEM (3.0)

```
WM_DELETEITEM  
idCtl = wParam; /* control identifier */  
lpdis = (const DELETEITEMSTRUCT FAR*) lParam; /* structure */
```

The WM\_DELETEITEM message is sent to the owner of an owner-drawn list box or combo box when the list box or combo box is destroyed or when items are removed by the LB\_DELETESTRING, LB\_RESETCONTENT, CB\_DELETESTRING, or CB\_RESETCONTENT message.

Parameter	Description
idCtl	Value of wParam. Specifies the identifier of the control that sent the WM_DELETEITEM message.
lpdis	Value of lParam. Points to a <u>DELETEITEMSTRUCT</u> structure that contains information about the item deleted from the list box.

Returns

An application should return TRUE if it processes this message.

See Also

CB\_DELETESTRING, CB\_RESETCONTENT, LB\_DELETESTRING, LB\_RESETCONTENT, DELETEITEMSTRUCT

### Windows 3.1 changes

The meaning of the `wParam` parameter has changed. The `wParam` parameter specifies the identifier of the control.

## WM\_DESTROY (2.x)

### WM\_DESTROY

The WM\_DESTROY message is sent when a window is being destroyed. It is sent to the window procedure of the window being destroyed after the window is removed from the screen.

This message is sent first to the window being destroyed and then to the child windows as they are destroyed. During the processing of the WM\_DESTROY message, it can be assumed that all child windows still exist.

### Parameters

This message has no parameters.

### Returns

An application should return zero if it processes this message.

### Comments

If the window being destroyed is part of the clipboard-viewer chain (set by calling the SetClipboardViewer function), the window must remove itself from the clipboard-viewer chain by calling the ChangeClipboardChain function before returning from the WM\_DESTROY message.

### Example

This example processes the WM\_DESTROY message by calling the PostQuitMessage function:

```
case WM_DESTROY:  
    PostQuitMessage(0);  
    return 0L;
```

### See Also

ChangeClipboardChain, DestroyWindow, PostQuitMessage, SetClipboardViewer, WM\_CLOSE

## WM\_DESTROYCLIPBOARD (2.x)

### WM\_DESTROYCLIPBOARD

The WM\_DESTROYCLIPBOARD message is sent to the clipboard owner when the clipboard is emptied by a call to the EmptyClipboard function.

#### Parameters

This message has no parameters.

#### Returns

An application should return zero if it processes this message.

#### See Also

EmptyClipboard



## WM\_DEVMODECHANGE (2.x)

WM\_DEVMODECHANGE

```
lpszDev = (LPCSTR) lParam; /* address of device name */
```

The WM\_DEVMODECHANGE message is sent to all top-level windows when the default device-mode settings have changed.

Parameter	Description
lpszDev	Value of lParam. Points to the device name specified in the Windows initialization file, WIN.INI.

### Returns

An application should return zero if it processes this message.

### Comments

Applications that receive this message may reinitialize their device-mode settings. Applications that use the ExtDeviceMode function to save and restore device settings typically do not process this message.

This message is not sent when the user changes the default printer from Control Panel. In this case, a WM\_WININICHANGE message is generated.

### See Also

ExtDeviceMode, WM\_WININICHANGE

## WM\_DRAWCLIPBOARD (2.x)

### WM\_DRAWCLIPBOARD

The WM\_DRAWCLIPBOARD message is sent to the first window in the clipboard-viewer chain when the contents of the clipboard change. Only applications that have joined the clipboard-viewer chain by calling the SetClipboardViewer function need to process this message.

#### Parameters

This message has no parameters.

#### Returns

An application should return zero if it processes this message.

#### Comments

Each window that receives the WM\_DRAWCLIPBOARD message should call the SendMessage function to pass the message on to the next window in the clipboard-viewer chain. The handle of the next window is returned by the SetClipboardViewer function; the handle may be modified in response to a WM\_CHANGECHAIN message.

#### See Also

SendMessage, SetClipboardViewer, WM\_CHANGECHAIN

## WM\_DRAWITEM (3.0)

```
WM_DRAWITEM
idCtl = (int) wParam; /* control identifier */
lpdis = (const DRAWITEMSTRUCT FAR*) lParam; /* structure*/
```

The WM\_DRAWITEM message is sent to the owner of an owner-drawn button, combo box, list box, or menu when a visual aspect of the button, combo box, list box, or menu has changed.

Parameter	Description
idCtl	Value of wParam. Specifies the identifier of the control that sent the WM_DRAWITEM message. This parameter is zero if the message was sent by a menu.
lpdis	Value of lParam. Points to a <u>DRAWITEMSTRUCT</u> structure that contains information about the item to be drawn and the type of drawing required.

### Returns

An application should return TRUE if it processes this message.

### Comments

The itemAction member of the DRAWITEMSTRUCT structure defines the drawing operation that is to be performed. The data in this member allows the owner of the control to determine what drawing action is required.

Before returning from processing this message, an application should ensure that the device context identified by the hDC member of the DRAWITEMSTRUCT structure is in the default state.

### Example

This example shows how to process the WM\_DRAWITEM message:

```
LPDRAWITEMSTRUCT lpdis;

case WM_DRAWITEM:
    lpdis = (DRAWITEMSTRUCT FAR*) lParam;

    switch (lpdis->itemAction) {

        case ODA_DRAWENTIRE:
            . /* Redraw the entire control or menu. */
            .
            return TRUE;

        case ODA_SELECT:
            . /* Redraw to reflect current selection state. */
            .
            return TRUE;

        case ODA_FOCUS:
            . /* Redraw to reflect current focus state. */
            .
            return TRUE;
    }
    break;
```

### See Also

WM\_COMPAREITEM, WM\_DELETEITEM, WM\_INITDIALOG, WM\_MEASUREITEM, DRAWITEMSTRUCT

### Windows 3.1 changes

The meaning of the `wParam` parameter has changed. If a control sends the `WM_DRAWITEM` message, the `wParam` parameter specifies the identifier of the control. If a menu sends the message, `wParam` is zero.

## WM\_DROPFILES (3.1)

### WM\_DROPFILES

```
hDrop = (HANDLE) wParam; /* handle of internal drop structure */
```

The WM\_DROPFILES message is sent when the user releases the left mouse button over the window of an application that has registered itself as a recipient of dropped files.

Parameter	Description
hDrop	Value of wParam. Identifies an internal data structure describing the dropped files. This handle is valid only during the processing of the WM_DROPFILES message; if an application needs to use the data later, it must allocate memory and save the data. This handle is used by the <u>DragFinish</u> , <u>DragQueryFile</u> , and <u>DragQueryPoint</u> functions to retrieve information about the dropped files.

### Returns

An application should return zero if it processes this message.

### Comments

This message is posted, not sent.

### See Also

DragAcceptFiles, DragFinish, DragQueryFile, DragQueryPoint

## WM\_ENABLE (2.x)

WM\_ENABLE

```
fEnabled = (BOOL) wParam; /* the enabled/disabled flag */
```

The WM\_ENABLE message is sent when an application changes the enabled state of a window. It is sent to the window whose enabled state is changing. This message is sent before the EnableWindow function returns but after the enabled state (WS\_DISABLE style bit) of the window has changed.

Parameter	Description
fEnabled	Value of wParam. Specifies whether the window has been enabled or disabled. This parameter is TRUE if the window has been enabled; it is FALSE if the window has been disabled.

### Returns

An application should return zero if it processes this message.

### See Also

EnableWindow

## WM\_ENDSESSION (2.x)

```
WM_ENDSESSION  
fEndSession = (BOOL) wParam;    /* end-session flag */
```

The WM\_ENDSESSION message is sent to an application that has returned a nonzero value in response to a WM\_QUERYENDSESSION message. The WM\_ENDSESSION message informs the application whether the session is actually ending.

Parameter	Description
fEndSession	Value of wParam. Specifies whether the session is being ended. It is TRUE if the session is being ended; otherwise, it is FALSE.

### Returns

An application should return zero if it processes this message.

### Comments

If the fEndSession parameter is TRUE, Windows can terminate any time after all applications have returned from processing this message. Therefore, an application should perform all tasks required for termination before returning from this message.

The application does not need to call the DestroyWindow or PostQuitMessage function when the session is ending.

### See Also

DestroyWindow, ExitWindows, PostQuitMessage, WM\_QUERYENDSESSION

## WM\_ENTERIDLE (2.x)

### WM\_ENTERIDLE

```
fwSource = wParam; /* idle-source flag*/  
hwndDlg = (HWND) LOWORD(lParam); /* handle of dialog box or window */  
/
```

The WM\_ENTERIDLE message informs an application's main window procedure that a modal dialog box or a menu is entering an idle state. A modal dialog box or menu enters an idle state when no messages are waiting in its queue after it has processed one or more previous messages.

Parameter	Description						
fwSource	Value of wParam. Specifies whether the message is the result of a dialog box or a menu being displayed. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>MSGF_DIALOGBOX</td><td>The system is idle because a dialog box is being displayed.</td></tr><tr><td>MSGF_MENU</td><td>The system is idle because a menu is being displayed.</td></tr></tbody></table>	Value	Description	MSGF_DIALOGBOX	The system is idle because a dialog box is being displayed.	MSGF_MENU	The system is idle because a menu is being displayed.
Value	Description						
MSGF_DIALOGBOX	The system is idle because a dialog box is being displayed.						
MSGF_MENU	The system is idle because a menu is being displayed.						
hwndDlg	Value of the low-order word of lParam. Identifies the dialog box (if fwSource is MSGF_DIALOGBOX) or the handle of the window containing the displayed menu (if fwSource is MSGF_MENU).						

### Returns

An application should return zero if it processes this message.

### Comments

The DefWindowProc function returns zero when it processes this message.

### See Also

DefWindowProc



## WM\_ERASEBKGD (2.x)

WM\_ERASEBKGD

```
hdc = (HDC) wParam; /* device-context handle */
```

The WM\_ERASEBKGD message is sent when the window background needs to be erased (for example, when a window is resized). It is sent to prepare an invalidated region for painting.

Parameter	Description
hdc	Value of wParam. Identifies the device context.

### Returns

An application should return nonzero if it erases the background; otherwise, it should return zero.

### Comments

The DefWindowProc function erases the background by using the class background brush specified by the hbrbackground member of the WNDCLASS structure.

If the hbrbackground member is NULL, the application should process the WM\_ERASEBKGD message and erase the background color. When processing the WM\_ERASEBKGD message, the application must align the origin of the intended brush with the window coordinates by first calling the UnrealizeObject function for the brush and then selecting the brush.

An application should return nonzero in response to WM\_ERASEBKGD if it processes the message and erases the background; this indicates that no further erasing is required. If the app returns zero the window will remain marked as needing to be erased. (Typically, this means that the fErase member of the PAINTSTRUCT structure will be TRUE.)

Windows computes the background by using the MM\_TEXT mapping mode. If the device context is using any other mapping mode, the area erased may not be within the visible part of the client area.

### See Also

UnrealizeObject, WM\_ICONERASEBKGD

## WM\_FONTCHANGE (2.x)

WM\_FONTCHANGE

```
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends the WM\_FONTCHANGE message to all top-level windows in the system after changing the pool of font resources.

Parameters

This message has no parameters.

Returns

An application should return zero if it processes this message.

Comments

An application that adds or removes fonts from the system (for example, by using the AddFontResource or RemoveFontResource function) should send this message to all top-level windows.

To send the WM\_FONTCHANGE message to all top-level windows, an application can call the SendMessage function with the hwnd parameter set to HWND\_BROADCAST.

See Also

AddFontResource, RemoveFontResource, SendMessage

## WM\_GETDLGCODE (2.x)

### WM\_GETDLGCODE

The WM\_GETDLGCODE message is sent to the dialog box procedure associated with a control. Normally, Windows handles all arrow-key and TAB-key input to the control. By responding to the WM\_GETDLGCODE message, an application can take control of a particular type of input and process the input itself.

#### Parameters

This message has no parameters.

#### Returns

The return value is any combination of the following values, indicating which type of input the application processes:

Value	Meaning
DLGC_BUTTON	Button (generic)
DLGC_DEFPUSHBUTTON	Default push button
DLGC_HASSETSEL	<u>EM_SETSEL</u> messages
DLGC_UNDEFPUSHBUTTON	No default push button processing. (An application can use this flag with DLGC_BUTTON to indicate that it processes button input but relies on the system for default push-button processing.)
DLGC_RADIOBUTTON	Radio button
DLGC_STATIC	Static control
DLGC_WANTALLKEYS	All keyboard input
DLGC_WANTARROWS	Arrow keys
DLGC_WANTCHARS	<u>WM_CHAR</u> messages
DLGC_WANTMESSAGE	All keyboard input (the application passes this message on to the control)
DLGC_WANTTAB	TAB key

#### Comments

Although the DefWindowProc function always returns zero in response to the WM\_GETDLGCODE message, the window procedures for the predefined control classes return a code appropriate for each class.

The WM\_GETDLGCODE message and the returned values are useful only with user-defined dialog box controls or standard controls modified by subclassing.

#### See Also

DefWindowProc

## WM\_GETFONT (3.0)

WM\_GETFONT

```
wParam = 0; /* not used, must be zero */
```

```
lParam = 0L; /* not used, must be zero */
```

An application sends a WM\_GETFONT message to a control to retrieve the font with which the control is currently drawing its text.

Parameters

This message has no parameters.

Returns

The return value is the handle of the font used by the control, or it is NULL if the control is using the system font.

See Also

[WM\\_SETFONT](#)

## WM\_GETMINMAXINFO (2.x)

WM\_GETMINMAXINFO

```
lpmmi = (MINMAXINFO FAR*) lParam; /* address of structure */
```

The WM\_GETMINMAXINFO message is sent to a window whenever Windows needs the maximized position or dimensions of the window or needs the maximum or minimum tracking size of the window. The maximized size of a window is the size of the window when its borders are fully extended. The maximum tracking size of a window is the largest window size that can be achieved by using the borders to size the window. The minimum tracking size of a window is the smallest window size that can be achieved by using the borders to size the window.

Windows fills in a MINMAXINFO data structure, specifying default values for the various positions and dimensions. The application may change these values if it processes this message.

Parameter	Description
lpmmi	Value of lParam. Points to a <u>MINMAXINFO</u> data structure.

### Returns

An application should return zero if it processes this message.

### Example

This example processes a WM\_GETMINMAXINFO message and sets the minimum tracking width of the window to 200 and the minimum tracking height of the window to 500:

```
MINMAXINFO FAR* lpmmi;  
  
case WM_GETMINMAXINFO:  
    lpmmi = (MINMAXINFO FAR*) lParam;  
    lpmmi->ptMinTrackSize.x = 200;  
    lpmmi->ptMinTrackSize.y = 500;  
  
    break;
```

### See Also

MINMAXINFO

## WM\_GETTEXT (2.x)

### WM\_GETTEXT

```
wParam = (WPARAM) cchTextMax; /* number of bytes to copy */  
lParam = (LPARAM) lpszText; /* address of buffer for text */
```

An application sends a WM\_GETTEXT message to copy the text that corresponds to a window into a buffer provided by the caller.

Parameter	Description
cchTextMax	Value of wParam. Specifies the maximum number of bytes to be copied, including the terminating null character.
lpszText	Value of lParam. Points to the buffer that is to receive the text.

### Returns

The return value is the number of bytes copied. It is CB\_ERR if the message is sent to a combo box that has no edit control.

### Comments

For an edit control, the text to be copied is the contents of the edit control. For a combo box, the text is the contents of the edit-control (or static-text) portion of the combo box. For a button, the text is the button name. For other windows, the text is the window title. To copy the text of an item in a list box, an application can use the LB\_GETTEXT message.

When the WM\_GETTEXT message is sent to a static control with the SS\_ICON style, the handle of the icon will be returned in the first two bytes of the buffer pointed to by lpszText. This is true only if the WM\_SETTEXT message has been used to set the icon.

### Example

This example copies text from an edit control to a buffer:

```
HWND hwndMyEdit;  
char szBuffer[32];  
  
hwndMyEdit = GetDlgItem(hdlg, ID_MYEDITCONTROL);  
SendMessage(hwndMyEdit, WM_GETTEXT, sizeof(szBuffer),  
            (LPARAM) ((LPSTR) szBuffer));
```

### See Also

LB\_GETTEXT, WM\_GETTEXTLENGTH, WM\_SETTEXT

## WM\_GETTEXTLENGTH (2.x)

### WM\_GETTEXTLENGTH

```
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends a WM\_GETTEXTLENGTH message to determine the length, in bytes, of the text associated with a window. The length does not include the terminating null character.

#### Parameters

This message has no parameters.

#### Returns

The return value is a word specifying the length, in bytes, of the text.

#### Comments

For an edit control, the text to be copied is the contents of the edit control. For a combo box, the text is the contents of the edit-control (or static-text) portion of the combo box. For a button, the text is the button name. For other windows, the text is the window title. To determine the length of an item in a list box, an application can use the LB\_GETTEXTLEN message.

#### Example

This example enables the push button in a dialog box if the user has entered text in an edit control in the dialog box:

```
case ID_MYEDITCONTROL:  
    if (HIWORD(lParam) == EN_CHANGE)  
        EnableWindow(GetDlgItem(hDlg, IDOK),  
        (BOOL) SendMessage(HWND) LOWORD(lParam),  
        WM_GETTEXTLENGTH, 0, 0L));  
    return TRUE;
```

#### See Also

LB\_GETTEXTLEN, WM\_GETTEXT

## WM\_HSCROLL (2.x)

```
WM_HSCROLL
wScrollCode = wParam; /* scroll bar code */
nPos = LOWORD(lParam); /* current position of scroll box */
hwndCtl = (HWND) HIWORD(lParam); /* handle of the control */
```

The WM\_HSCROLL message is sent to a window when the user clicks the window's horizontal scroll bar.

Parameter	Description																				
wScrollCode	Value of wParam. Specifies a scroll bar code that indicates the user's scrolling request. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>SB_BOTTOM</td><td>Scroll to bottom.</td></tr><tr><td>SB_ENDSCROLL</td><td>End scroll.</td></tr><tr><td>SB_LINEDOWN</td><td>Scroll one line down.</td></tr><tr><td>SB_LINEUP</td><td>Scroll one line up.</td></tr><tr><td>SB_PAGEDOWN</td><td>Scroll one page down.</td></tr><tr><td>SB_PAGEUP</td><td>Scroll one page up.</td></tr><tr><td>SB_THUMBPOSITION</td><td>Scroll to absolute position. The current position is specified by the nPos parameter.</td></tr><tr><td>SB_THUMBTRACK</td><td>Drag scroll box (thumb) to specified position. The current position is specified by the nPos parameter.</td></tr><tr><td>SB_TOP</td><td>Scroll to top.</td></tr></tbody></table>	Value	Description	SB_BOTTOM	Scroll to bottom.	SB_ENDSCROLL	End scroll.	SB_LINEDOWN	Scroll one line down.	SB_LINEUP	Scroll one line up.	SB_PAGEDOWN	Scroll one page down.	SB_PAGEUP	Scroll one page up.	SB_THUMBPOSITION	Scroll to absolute position. The current position is specified by the nPos parameter.	SB_THUMBTRACK	Drag scroll box (thumb) to specified position. The current position is specified by the nPos parameter.	SB_TOP	Scroll to top.
Value	Description																				
SB_BOTTOM	Scroll to bottom.																				
SB_ENDSCROLL	End scroll.																				
SB_LINEDOWN	Scroll one line down.																				
SB_LINEUP	Scroll one line up.																				
SB_PAGEDOWN	Scroll one page down.																				
SB_PAGEUP	Scroll one page up.																				
SB_THUMBPOSITION	Scroll to absolute position. The current position is specified by the nPos parameter.																				
SB_THUMBTRACK	Drag scroll box (thumb) to specified position. The current position is specified by the nPos parameter.																				
SB_TOP	Scroll to top.																				
nPos	Value of the low-order word of lParam. Specifies the current position of the scroll box if the wScrollCode parameter is SB_THUMBPOSITION or SB_THUMBTRACK; otherwise, the nPos parameter is not used.																				
hwndCtl	Value of the high-order word of lParam. Identifies the control if WM_HSCROLL is sent by a scroll bar. If WM_HSCROLL is sent as a result of the user clicking a pop-up window's scroll bar, the high-order word is not used.																				

### Returns

An application should return zero if it processes this message.

### Comments

The SB\_THUMBTRACK scroll bar code typically is used by applications that give some feedback while the scroll box is being dragged.

If an application scrolls the contents of the window, it must also reset the position of the scroll box by using the [SetScrollPos](#) function.

### See Also

[SetScrollPos](#), [WM\\_VSCROLL](#)



## WM\_HSCROLLCLIPBOARD (2.x)

```
WM_HSCROLLCLIPBOARD
hwndCBViewer = (HWND) wParam; /* handle of clipboard viewer */
wScrollCode = LOWORD(lParam); /* scroll bar code */
nPos = (int) HIWORD(lParam); /* scroll box position */
```

The WM\_HSCROLLCLIPBOARD message is sent by the clipboard viewer to the clipboard owner when the clipboard data has the CF\_OWNERDISPLAY format and an event occurs in the clipboard viewer's horizontal scroll bar. The owner should scroll the clipboard image, invalidate the appropriate section, and update the scroll bar values.

Parameter	Description																		
hwndCBViewer	Value of wParam. Identifies a clipboard-viewer window.																		
wScrollCode	Value of the low-order word of lParam. Specifies a scroll bar code. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>SB_BOTTOM</td><td>Scroll to lower right.</td></tr><tr><td>SB_ENDSCROLL</td><td>End scroll.</td></tr><tr><td>SB_LINEDOWN</td><td>Scroll one line down.</td></tr><tr><td>SB_LINEUP</td><td>Scroll one line up.</td></tr><tr><td>SB_PAGEDOWN</td><td>Scroll one page down.</td></tr><tr><td>SB_PAGEUP</td><td>Scroll one page up.</td></tr><tr><td>SB_THUMBPOSITION</td><td>Scroll to absolute position.</td></tr><tr><td>SB_TOP</td><td>Scroll to upper left.</td></tr></tbody></table>	Value	Description	SB_BOTTOM	Scroll to lower right.	SB_ENDSCROLL	End scroll.	SB_LINEDOWN	Scroll one line down.	SB_LINEUP	Scroll one line up.	SB_PAGEDOWN	Scroll one page down.	SB_PAGEUP	Scroll one page up.	SB_THUMBPOSITION	Scroll to absolute position.	SB_TOP	Scroll to upper left.
Value	Description																		
SB_BOTTOM	Scroll to lower right.																		
SB_ENDSCROLL	End scroll.																		
SB_LINEDOWN	Scroll one line down.																		
SB_LINEUP	Scroll one line up.																		
SB_PAGEDOWN	Scroll one page down.																		
SB_PAGEUP	Scroll one page up.																		
SB_THUMBPOSITION	Scroll to absolute position.																		
SB_TOP	Scroll to upper left.																		
nPos	Value of the high-order word of lParam. Specifies the scroll box position if the scroll bar code is SB_THUMBPOSITION; otherwise, the high-order word of lParam is not used.																		

### Returns

An application should return zero if it processes this message.

### Comments

The clipboard owner should use the [InvalidateRect](#) function or repaint as needed. The scroll bar position should also be reset.

### See Also

[InvalidateRect](#), [WM\\_VSCROLLCLIPBOARD](#)

## WM\_ICONERASEBKGND (3.0)

WM\_ICONERASEBKGND

```
hdc = (HDC) wParam; /* device-context handle */
```

The WM\_ICONERASEBKGND message is sent to a minimized (iconic) window when the background of the icon must be filled before painting the icon. A window receives this message only if a class icon is defined for the window; otherwise, WM\_ERASEBKGND is sent.

Parameter	Description
hdc	Value of wParam. Identifies the device context of the icon.

### Returns

An application should return zero if it processes this message.

### Comments

The DefWindowProc function fills the icon background with the background brush of the parent window.

### See Also

DefWindowProc, WM\_ERASEBKGND

## WM\_INITDIALOG (2.x)

WM\_INITDIALOG

```
hwndFocus = (HWND) wParam; /* handle of control for focus */
dwData = lParam; /* application-specific data */
```

The WM\_INITDIALOG message is sent to a dialog box procedure immediately before the dialog box is displayed.

Parameter	Description
hwndFocus	Value of wParam. Identifies the first control in the dialog box that can be given the input focus. Usually, this is the first control in the dialog box with the <u>WS_TABSTOP</u> style.
dwData	Value of lParam. Specifies application-specific data that was passed by the function used to create the dialog box if the dialog box was created by one of the following functions:

CreateDialogParam  
DialogBoxIndirectParam  
DialogBoxParam

### Returns

An application should return nonzero to set the input focus to the control identified by the hwndFocus parameter. An application should return zero if the dialog box procedure uses the SetFocus function to set the input focus to a different control in the dialog box.

### Example

This example changes the font used by controls in a dialog box to a font that is not bold.

```
HFONT hfontDlg;
LOGFONT lFont;

case WM_INITDIALOG:

    /* Get dialog box font and create version that is not bold. */

    hfontDlg = (HFONT) NULL;
    if ((hfontDlg = (HFONT) SendMessage(hdlg, WM_GETFONT, 0, 0L))) {
        if (GetObject(hfontDlg, sizeof(LOGFONT), (LPSTR) &lFont)) {
            lFont.lfWeight = FW_NORMAL;
            if (hfontDlg = CreateFontIndirect((LPLOGFONT) &lFont)) {
                SendDlgItemMessage(hdlg, ID_CTRL1, WM_SETFONT,
(WPARAM) hfontDlg, 0);
                SendDlgItemMessage(hdlg, ID_CTRL2, WM_SETFONT,
(WPARAM) hfontDlg, 0);
            }
        }
        /* Set font for remaining controls. */
    }
}
return TRUE;
```

### See Also

CreateDialogParam, DialogBoxIndirectParam, DialogBoxParam, SetFocus

## WM\_INITMENU (2.x)

WM\_INITMENU

```
hmenuInit = (HMENU) wParam; /* handle of menu to initialize */
```

The WM\_INITMENU message is sent when a menu is about to become active. It occurs when the user clicks an item on the menu bar or presses a menu key. This allows an application to modify the menu before it is displayed.

Parameter	Description
hmenuInit	Value of wParam. Identifies the menu to be initialized.

### Returns

An application should return zero if it processes this message.

### Comments

This message is sent only when a menu is first accessed; only one WM\_INITMENU message is generated for each access. This means, for example, that moving the mouse across several menu items while holding down the button does not generate new messages. WM\_INITMENU does not provide information about menu items.

### See Also

WM\_INITMENUPOPUP

## WM\_INITMENUPOPUP (2.x)

```
WM_INITMENUPOPUP
hmenuPopup = (HMENU) wParam;    /* handle of pop-up menu */
nIndex = (int) LOWORD(lParam);  /* index of pop-up menu */
fSystemMenu = (BOOL) HIWORD(lParam); /* System-menu flag */
```

The WM\_INITMENUPOPUP message is sent when a pop-up menu is about to become active. This allows an application to modify the pop-up menu before it is displayed, without changing the entire menu.

Parameter	Description
hmenuPopup	Value of wParam. Identifies the pop-up menu.
nIndex	Value of the low-order word of lParam. Specifies the index of the pop-up menu in the main menu.
fSystemMenu	Value of the high-order word of lParam. Specifies a nonzero value if the pop-up menu is the System menu (sometimes referred to as the Control menu); otherwise, this parameter is zero.

### Returns

An application should return zero if it processes this message.

### Example

This example initializes the items in a pop-up menu:

```
int  cItems;
int  pos;
UINT id;

case WM_INITMENUPOPUP:
    cItems = GetMenuItemCount((HMENU) wParam);
    for (pos = 0; pos < cItems; pos++) {
        id = GetMenuItemID((HMENU) wParam, pos);
        . /* Initialize menu items. */
        .
    }
    break;
```

### See Also

WM\_INITMENU

## WM\_KEYDOWN (2.x)

```
WM_KEYDOWN
wVkey = wParam; /* virtual-key code */
dwKeyData = lParam; /* key data */
```

The WM\_KEYDOWN message is sent when a nonsystem key is pressed. A nonsystem key is a key that is pressed when the ALT key is not pressed, or it is a key that is pressed when a window has the input focus.

Parameter	Description																		
wVkey	Value of wParam. Specifies the virtual-key code of the given key.																		
dwKeydata	Value of lParam. Specifies the repeat count, scan code, extended key, context code, previous key state, and key-transition state, as shown in the following table:																		
	<table><thead><tr><th>Bit</th><th>Description</th></tr></thead><tbody><tr><td>0-15</td><td>Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.</td></tr><tr><td>16-23</td><td>Specifies the scan code. The value depends on the original equipment manufacturer (OEM).</td></tr><tr><td>24</td><td>Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.</td></tr><tr><td>25-26</td><td>Not used.</td></tr><tr><td>27-28</td><td>Used internally by Windows.</td></tr><tr><td>29</td><td>Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.</td></tr><tr><td>30</td><td>Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.</td></tr><tr><td>31</td><td>Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.</td></tr></tbody></table>	Bit	Description	0-15	Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.	16-23	Specifies the scan code. The value depends on the original equipment manufacturer (OEM).	24	Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.	25-26	Not used.	27-28	Used internally by Windows.	29	Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.	30	Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.	31	Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.
Bit	Description																		
0-15	Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.																		
16-23	Specifies the scan code. The value depends on the original equipment manufacturer (OEM).																		
24	Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.																		
25-26	Not used.																		
27-28	Used internally by Windows.																		
29	Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.																		
30	Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.																		
31	Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.																		

For a WM\_KEYDOWN message, the value of bit 29 (context code) is 0 and the value of bit 31 (key-transition state) is 0.

### Returns

An application should return zero if it processes this message.

### Comments

Because of the autorepeat feature, more than one WM\_KEYDOWN message may occur before a WM\_KEYUP message is sent. The previous key state (bit 30) can be used to determine whether the WM\_KEYDOWN message indicates the first down transition or a repeated down transition.

For IBM Enhanced 101- and 102-key keyboards, enhanced keys are the right ALT key and the right CTRL key on the main section of the keyboard; the INS, DEL, HOME, END, PAGE UP, PAGE DOWN, and arrow keys in the clusters to the left of the numeric keypad; and the division (/) and ENTER key on the numeric keypad. Some other keyboards may support the extended-key bit in the dwKeyData parameter.

### See Also

WM\_CHAR, WM\_KEYUP

## WM\_KEYUP (2.x)

```
WM_KEYUP
wVkey = wParam; /* virtual-key code */
dwKeyData = lParam; /* key data */
```

The WM\_KEYUP message is sent when a nonsystem key is released. A nonsystem key is a key that is pressed when the ALT key is not pressed, or it is a key that is pressed when a window has the input focus.

Parameter	Description																		
wVkey	Value of wParam. Specifies the virtual-key code of the given key.																		
dwKeyData	Value of lParam. Specifies the repeat count, scan code, extended key, context code, previous key state, and key-transition state, as shown in the following table:																		
	<table><thead><tr><th>Bit</th><th>Description</th></tr></thead><tbody><tr><td>0-15</td><td>Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.</td></tr><tr><td>16-23</td><td>Specifies the scan code. The value depends on the original equipment manufacturer (OEM).</td></tr><tr><td>24</td><td>Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.</td></tr><tr><td>25-26</td><td>Not used.</td></tr><tr><td>27-28</td><td>Used internally by Windows.</td></tr><tr><td>29</td><td>Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.</td></tr><tr><td>30</td><td>Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.</td></tr><tr><td>31</td><td>Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.</td></tr></tbody></table>	Bit	Description	0-15	Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.	16-23	Specifies the scan code. The value depends on the original equipment manufacturer (OEM).	24	Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.	25-26	Not used.	27-28	Used internally by Windows.	29	Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.	30	Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.	31	Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.
Bit	Description																		
0-15	Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.																		
16-23	Specifies the scan code. The value depends on the original equipment manufacturer (OEM).																		
24	Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.																		
25-26	Not used.																		
27-28	Used internally by Windows.																		
29	Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.																		
30	Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.																		
31	Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.																		
	For a WM_KEYUP message, the value of bit 29 (context code) is 0 and the value of bit 31 (key-transition state) is 1.																		

### Returns

An application should return zero if it processes this message.

### Comments

For IBM Enhanced 101- and 102-key keyboards, enhanced keys are the right ALT key and the right CTRL key on the main section of the keyboard; the INS, DEL, HOME, END, PAGE UP, PAGE DOWN, and arrow keys in the clusters to the left of the numeric keypad; and the division (/) and ENTER keys on the numeric keypad. Some other keyboards may support the extended-key bit in the dwKeyData parameter.

### See Also

WM\_CHAR, WM\_KEYDOWN

## WM\_KILLFOCUS (2.x)

WM\_KILLFOCUS

```
hwndGetFocus = (HWND) lParam; /* handle of window receiving focus */
```

The WM\_KILLFOCUS message is sent immediately before a window loses the input focus.

Parameter	Description
hwndGetFocus	Value of wParam. Identifies the window that receives the input focus. (This parameter may be NULL.)

### Returns

An application should return zero if it processes this message.

### Comments

If an application is displaying a caret, the caret should be destroyed at this point.

### See Also

SetFocus, WM\_SETFOCUS



## WM\_LBUTTONDOWNBLCLK (2.x)

```
WM_LBUTTONDOWNBLCLK
fwKeys = wParam; /* key flags */
xPos = LOWORD(lParam); /* horizontal position of cursor */
yPos = HIWORD(lParam); /* vertical position of cursor */
```

The WM\_LBUTTONDOWNBLCLK message is sent when the user double-clicks the left mouse button.

Parameter	Description												
fwKeys	Value of wParam. Indicates whether various virtual keys are down. This parameter can be any combination of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>MK_CONTROL</td><td>Set if CTRL key is down.</td></tr><tr><td>MK_LBUTTON</td><td>Set if left button is down.</td></tr><tr><td>MK_MBUTTON</td><td>Set if middle button is down.</td></tr><tr><td>MK_RBUTTON</td><td>Set if right button is down.</td></tr><tr><td>MK_SHIFT</td><td>Set if SHIFT key is down.</td></tr></tbody></table>	Value	Description	MK_CONTROL	Set if CTRL key is down.	MK_LBUTTON	Set if left button is down.	MK_MBUTTON	Set if middle button is down.	MK_RBUTTON	Set if right button is down.	MK_SHIFT	Set if SHIFT key is down.
Value	Description												
MK_CONTROL	Set if CTRL key is down.												
MK_LBUTTON	Set if left button is down.												
MK_MBUTTON	Set if middle button is down.												
MK_RBUTTON	Set if right button is down.												
MK_SHIFT	Set if SHIFT key is down.												
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.												
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.												

### Returns

An application should return zero if it processes this message.

### Comments

Only windows that have the CS\_DBLCLKS class style can receive WM\_LBUTTONDOWNBLCLK messages. Windows generates a WM\_LBUTTONDOWNBLCLK message when the user presses, releases, and again presses the left mouse button within the system's double-click time limit. Double-clicking the left mouse button actually generates four messages: a WM\_LBUTTONDOWN message, a WM\_LBUTTONUP message, the WM\_LBUTTONDOWNBLCLK message, and another WM\_LBUTTONUP message.

### See Also

WM\_LBUTTONDOWN, WM\_LBUTTONUP

## WM\_LBUTTONDOWN (2.x)

### WM\_LBUTTONDOWN

```
fwKeys = wParam; /* key flags */
xPos = LOWORD(lParam); /* horizontal position of cursor */
yPos = HIWORD(lParam); /* vertical position of cursor */
```

The WM\_LBUTTONDOWN message is sent when the user presses the left mouse button.

Parameter	Description										
fwKeys	Value of wParam. Specifies whether various virtual keys are down. This parameter can be any combination of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>MK_CONTROL</td><td>Set if CTRL key is down.</td></tr><tr><td>MK_MBUTTON</td><td>Set if middle button is down.</td></tr><tr><td>MK_RBUTTON</td><td>Set if right button is down.</td></tr><tr><td>MK_SHIFT</td><td>Set if SHIFT key is down.</td></tr></tbody></table>	Value	Description	MK_CONTROL	Set if CTRL key is down.	MK_MBUTTON	Set if middle button is down.	MK_RBUTTON	Set if right button is down.	MK_SHIFT	Set if SHIFT key is down.
Value	Description										
MK_CONTROL	Set if CTRL key is down.										
MK_MBUTTON	Set if middle button is down.										
MK_RBUTTON	Set if right button is down.										
MK_SHIFT	Set if SHIFT key is down.										
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.										
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.										

### Returns

An application should return zero if it processes this message.

### See Also

WM\_LBUTTONDOWNBLCLK, WM\_LBUTTONUP

## WM\_LBUTTONDOWN (2.x)

### WM\_LBUTTONDOWN

```
fwKeys = wParam; /* key flags */
xPos = LOWORD(lParam); /* horizontal position of cursor */
yPos = HIWORD(lParam); /* vertical position of cursor */
```

The WM\_LBUTTONDOWN message is sent when the user releases the left mouse button.

Parameter	Description										
fwKeys	Value of wParam. Indicates whether various virtual keys are down. This parameter can be any combination of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>MK_CONTROL</td><td>Set if CTRL key is down.</td></tr><tr><td>MK_MBUTTON</td><td>Set if middle button is down.</td></tr><tr><td>MK_RBUTTON</td><td>Set if right button is down.</td></tr><tr><td>MK_SHIFT</td><td>Set if SHIFT key is down.</td></tr></tbody></table>	Value	Description	MK_CONTROL	Set if CTRL key is down.	MK_MBUTTON	Set if middle button is down.	MK_RBUTTON	Set if right button is down.	MK_SHIFT	Set if SHIFT key is down.
Value	Description										
MK_CONTROL	Set if CTRL key is down.										
MK_MBUTTON	Set if middle button is down.										
MK_RBUTTON	Set if right button is down.										
MK_SHIFT	Set if SHIFT key is down.										
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.										
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.										

### Returns

An application should return zero if it processes this message.

### See Also

[WM\\_LBUTTONDOWNBLCLK](#), [WM\\_LBUTTONDOWNDOWN](#)

## WM\_MBUTTONDOWNBLCLK (2.x)

```
WM_MBUTTONDOWNBLCLK
fwKeys = wParam; /* key flags */
xPos = LOWORD(lParam); /* horizontal position of cursor */
yPos = HIWORD(lParam); /* vertical position of cursor */
```

The WM\_MBUTTONDOWNBLCLK message is sent when the user double-clicks the middle mouse button.

Parameter	Description												
fwKeys	Value of wParam. Indicates whether various virtual keys are down. This parameter can be any combination of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>MK_CONTROL</td><td>Set if CTRL key is down.</td></tr><tr><td>MK_LBUTTON</td><td>Set if left button is down.</td></tr><tr><td>MK_MBUTTON</td><td>Set if middle button is down.</td></tr><tr><td>MK_RBUTTON</td><td>Set if right button is down.</td></tr><tr><td>MK_SHIFT</td><td>Set if SHIFT key is down.</td></tr></tbody></table>	Value	Description	MK_CONTROL	Set if CTRL key is down.	MK_LBUTTON	Set if left button is down.	MK_MBUTTON	Set if middle button is down.	MK_RBUTTON	Set if right button is down.	MK_SHIFT	Set if SHIFT key is down.
Value	Description												
MK_CONTROL	Set if CTRL key is down.												
MK_LBUTTON	Set if left button is down.												
MK_MBUTTON	Set if middle button is down.												
MK_RBUTTON	Set if right button is down.												
MK_SHIFT	Set if SHIFT key is down.												
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.												
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.												

### Returns

An application should return zero if it processes this message.

### Comments

Only windows that have the CS\_DBLCLKS class style can receive WM\_MBUTTONDOWNBLCLK messages. Windows generates a WM\_MBUTTONDOWNBLCLK message when the user presses, releases, and again presses the middle mouse button within the system's double-click time limit. Double-clicking the middle mouse button actually generates four messages: a WM\_MBUTTONDOWN message, a WM\_MBUTTONUP message, the WM\_MBUTTONDOWNBLCLK message, and another WM\_MBUTTONUP message.

### See Also

WM\_MBUTTONDOWN, WM\_MBUTTONUP

## WM\_MBUTTONDOWN (2.x)

WM\_MBUTTONDOWN

```
fwKeys = wParam; /* key flags */
xPos = LOWORD(lParam); /* horizontal position of cursor */
yPos = HIWORD(lParam); /* vertical position of cursor */
```

The WM\_MBUTTONDOWN message is sent when the user presses the middle mouse button.

Parameter	Description										
fwKeys	Value of wParam. Indicates whether various virtual keys are down. This parameter can be any combination of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>MK_CONTROL</td><td>Set if CTRL key is down.</td></tr><tr><td>MK_LBUTTON</td><td>Set if left button is down.</td></tr><tr><td>MK_RBUTTON</td><td>Set if right button is down.</td></tr><tr><td>MK_SHIFT</td><td>Set if SHIFT key is down.</td></tr></tbody></table>	Value	Description	MK_CONTROL	Set if CTRL key is down.	MK_LBUTTON	Set if left button is down.	MK_RBUTTON	Set if right button is down.	MK_SHIFT	Set if SHIFT key is down.
Value	Description										
MK_CONTROL	Set if CTRL key is down.										
MK_LBUTTON	Set if left button is down.										
MK_RBUTTON	Set if right button is down.										
MK_SHIFT	Set if SHIFT key is down.										
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.										
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.										

### Returns

An application should return zero if it processes this message.

### See Also

[WM\\_MBUTTONDOWNBLCLK](#), [WM\\_MBUTTONUP](#)

## WM\_MBUTTONDOWN (2.x)

### WM\_MBUTTONDOWN

```
fwKeys = wParam; /* key flags */  
xPos = LOWORD(lParam); /* horizontal position of cursor */  
yPos = HIWORD(lParam); /* vertical position of cursor */
```

The WM\_MBUTTONDOWN message is sent when the user releases the middle mouse button.

Parameter	Description										
fwKeys	Value of wParam. Indicates whether various virtual keys are down. This parameter can be any combination of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>MK_CONTROL</td><td>Set if CTRL key is down.</td></tr><tr><td>MK_LBUTTON</td><td>Set if left button is down.</td></tr><tr><td>MK_RBUTTON</td><td>Set if right button is down.</td></tr><tr><td>MK_SHIFT</td><td>Set if SHIFT key is down.</td></tr></tbody></table>	Value	Description	MK_CONTROL	Set if CTRL key is down.	MK_LBUTTON	Set if left button is down.	MK_RBUTTON	Set if right button is down.	MK_SHIFT	Set if SHIFT key is down.
Value	Description										
MK_CONTROL	Set if CTRL key is down.										
MK_LBUTTON	Set if left button is down.										
MK_RBUTTON	Set if right button is down.										
MK_SHIFT	Set if SHIFT key is down.										
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.										
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.										

### Returns

An application should return zero if it processes this message.

### See Also

[WM\\_MBUTTONDOWNBLCLK](#), [WM\\_MBUTTONDOWN](#)

## WM\_MDIACTIVATE (3.0)

```
WM_MDIACTIVATE
/* Message sent to MDI client */
wParam = (WPARAM) (HWND) hwndChildAct; /* child to activate */
lParam = 0L; /* not used, must be zero */

/* Message received by MDI child */
wParam = (WPARAM) fActivate; /* activation flag */
hwndAct = (HWND) LOWORD(lParam); /* child being activated */
hwndDeact = (HWND) HIWORD(lParam); /* child being deactivated */
```

An application sends the WM\_MDIACTIVATE message to a multiple document interface (MDI) client window to instruct the client window to activate a different MDI child window. As the client window processes this message, it sends WM\_MDIACTIVATE to the child window being deactivated and to the child window being activated.

Parameter	Description
In message sent to MDI client window: hwndChildAct	Value of wParam. Identifies the MDI child window to be activated.
In message received by MDI child window: fActivate	Value of wParam. Specifies whether to activate or deactivate the child window. If this parameter is TRUE, the child window is activated. If this parameter is FALSE, the child window is deactivated.
hwndAct	Value of the low-order word of lParam. Identifies the child window being activated.
hwndDeact	Value of the high-order word of lParam. Identifies the child window being deactivated.

### Returns

An application should return zero if it processes this message.

### Comments

An MDI child window is activated independently of the MDI frame window. When the frame window becomes active, the child window that was last activated with the WM\_MDIACTIVATE message receives the WM\_NCACTIVATE message to draw an active window frame and title bar; it does not receive another WM\_MDIACTIVATE message.

### See Also

[WM\\_MDIGETACTIVE](#), [WM\\_NCACTIVATE](#), [WM\\_MDINEXT](#)

■ **WM\_MDICASCADE (3.0)**

WM\_MDICASCADE

```
fnCascade = wParam; /* cascade flag */
```

The WM\_MDICASCADE message is sent to a multiple document interface (MDI) client window to arrange all its child windows in a cascade format.

Parameter	Description				
fnCascade	Value of wParam. Specifies a cascade flag. Currently, only the following flag may be specified:				
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MDITILE_SKIPDISABLED</td><td>Prevents disabled MDI child windows from being cascaded.</td></tr></tbody></table>	Value	Meaning	MDITILE_SKIPDISABLED	Prevents disabled MDI child windows from being cascaded.
Value	Meaning				
MDITILE_SKIPDISABLED	Prevents disabled MDI child windows from being cascaded.				

**Returns**

An application should return zero if it processes this message.

**See Also**

WM\_MDIICONARRANGE, WM\_MDITILE



### Windows 3.1 changes

The following cascade flag has been added:

Value	Meaning
MDITILE_SKIPDISABLED	Prevents disabled MDI child windows from being cascaded.

## WM\_MDICREATE (3.0)

```
WM_MDICREATE
wParam = 0; /* not used, must be zero */
lParam = (LPARAM) (MDICREATESTRUCT FAR*) lpms; /* structure address *
/
```

An application sends the WM\_MDICREATE message to a multiple document interface (MDI) client window to create a child window.

Parameter	Description
lpms	Value of lParam. Points to an <u>MDICREATESTRUCT</u> structure.

### Returns

The return value is the handle of the new window in the low-order word and zero in the high-order word.

### Comments

The window is created with the style bits WS\_CHILD, WS\_CLIPSIBLINGS, WS\_CLIPCHILDREN, WS\_SYSMENU, WS\_CAPTION, WS\_THICKFRAME, WS\_MINIMIZEBOX, and WS\_MAXIMIZEBOX, plus additional style bits specified in the MDICREATESTRUCT structure to which lpms points. Windows adds the title of the new child window to the window menu of the frame window. An application should create all child windows of the client window with this message.

CreateWindow will override the default style bits if the MDIS\_ALLCHILDSTYLES style is set when creating the MDI client window.

If a client window receives any message that changes the activation of child windows while the currently active MDI child window is maximized, Windows restores the currently active child window and maximizes the newly activated child window.

When the MDI child window is created, Windows sends the WM\_CREATE message to the window. The lpms parameter of the WM\_CREATE message contains a pointer to a CREATESTRUCT structure. The lpCreateParams member of the CREATESTRUCT structure contains a pointer to the MDICREATESTRUCT structure passed with the WM\_MDICREATE message that created the MDI child window.

An application should not send a second WM\_MDICREATE message while a WM\_MDICREATE message is still being processed. For example, it should not send a WM\_MDICREATE message while an MDI child window is processing its WM\_CREATE message.

### See Also

WM\_MDIDESTROY, CREATESTRUCT, MDICREATESTRUCT

## WM\_MDIDESTROY (3.0)

WM\_MDIDESTROY

```
hwndChild = (HWND) wParam; /* handle of child to destroy */
```

An application sends the WM\_MDIDESTROY message to a multiple document interface (MDI) client window to close an MDI child window.

Parameter	Description
hwndChild	Value of wParam. Identifies the child window to destroy.

### Returns

An application should return zero if it processes this message.

### Comments

This message removes the title of the child window from the frame window and deactivates the child window. An application should close all MDI child windows with this message.

If a client window receives any message that changes the activation of child windows while the currently active MDI child window is maximized, Windows restores the currently active child window and maximizes the newly activated child window.

See Also

WM\_MDICREATE

## WM\_MDIGETACTIVE (3.0)

### WM\_MDIGETACTIVE

The WM\_MDIGETACTIVE message retrieves the multiple document interface (MDI) child window that is active, along with a flag indicating whether the child window is maximized.

#### Parameters

This message has no parameters.

#### Returns

The return value is the handle of the active MDI child window in its low-order word. If the window is maximized, the high-order word is 1; otherwise, the high-order word is 0.

#### See Also

WM\_MDIACTIVATE

## WM\_MDIICONARRANGE (3.0)

### WM\_MDIICONARRANGE

The WM\_MDIICONARRANGE message is sent to a multiple document interface (MDI) client window to arrange all minimized document child windows. It does not affect child windows that are not minimized.

#### Parameters

This message has no parameters.

#### Returns

An application should return zero if it processes this message.

#### See Also

WM\_MDICASCADE, WM\_MDITILE

## WM\_MDIMAXIMIZE (3.0)

WM\_MDIMAXIMIZE

```
hwndMaximize = (HWND) wParam; /* handle of child to maximize */
```

The WM\_MDIMAXIMIZE message causes a multiple document interface (MDI) client window to maximize an MDI child window. When a child window is maximized, Windows resizes it to make its client area fill the client window. Windows places the child window's System menu (sometimes referred to as the Control menu) in the frame's menu bar so that the user can restore or minimize the child window; Windows adds the title of the child window to the frame window's menu of child windows.

Parameter	Description
hwndMaximize	Value of wParam. Identifies the child window to maximize.

### Returns

An application should return zero if it processes this message.

### Comments

If an MDI client window receives any message that changes the activation of its child windows while the currently active MDI child window is maximized, Windows restores the currently active child window and maximizes the newly activated child window.

## WM\_MDINEXT (3.0)

### WM\_MDINEXT

```
wParam = (WPARAM) hwndChild; /* handle of child window */  
lParam = (LPARAM) fNext; /* next or previous child window */
```

An application sends the WM\_MDINEXT message to a multiple document interface (MDI) client window to activate the child window immediately behind the currently active child window and place the currently active child window behind all other child windows.

Parameter	Description
hwndChild	Value of wParam. Specifies the handle of the child window.
fNext	Value of lParam. If this parameter is zero, the message specifies that the next MDI child window should be activated. If this parameter is nonzero, the message specifies that the previous MDI child window should be activated.

### Returns

An application should return zero if it processes this message.

### Comments

If an MDI client window receives any message that changes the activation of its child windows while the currently active MDI child window is maximized, Windows restores the currently active child window and maximizes the newly activated child window.

### See Also

WM\_MDIACTIVATE, WM\_MDIGETACTIVE

## WM\_MDIRESTORE (3.0)

WM\_MDIRESTORE

```
wParam = (WPARAM) wIDChild; /* handle of child window */
```

An application sends the WM\_MDIRESTORE message to a multiple document interface (MDI) client window to restore an MDI child window from maximized or minimized size.

Parameter	Description
wIDChild	Value of wParam. Specifies the handle of the child window.

Returns

An application should return zero if it processes this message.

See Also

WM\_MDIMAXIMIZE



## WM\_MDISETMENU (3.0)

WM\_MDISETMENU

```
wParam = (WPARAM) (BOOL) fRefresh; /* refresh flag */  
lParam = MAKELPARAM(hmenuFrame, hmenuWindow); /* new menus */
```

An application sends a WM\_MDISETMENU message to replace the menu of a multiple document interface (MDI) frame window, the Window pop-up menu, or both.

Parameter	Description
fRefresh	Value of wParam. Specifies whether to refresh the current menus or specify new menus. It is TRUE if the menus should just be refreshed. It is FALSE if, instead, the hmenuFrame and hmenuWindow parameters should be used to specify new menus for the window.
hmenuFrame	Value of the low-order word of lParam. Identifies the new frame-window menu. If this parameter is zero, the frame-window menu is not changed.
hmenuWindow	Value of the high-order word of lParam. Identifies the new Window pop-up menu. If this parameter is zero, the Window pop-up menu is not changed.

### Returns

The return value is the handle of the frame-window menu replaced by this message.

### Comments

After sending this message, an application must call the DrawMenuBar function to update the menu bar.

If this message replaces the Window pop-up menu, MDI child-window menu items are removed from the previous Window menu and added to the new Window pop-up menu.

If an MDI child window is maximized and this message replaces the MDI frame-window menu, the System menu (sometimes referred to as the Control menu) and restore controls are removed from the previous frame-window menu and added to the new menu.

### See Also

DrawMenuBar

### Windows 3.1 changes

The wParam parameter specifies whether or not to refresh the same menus. It is nonzero if the menus should just be refreshed. It is zero if, instead, the lParam parameter should be used to specify new menus for the window.

The previous release of the documentation stated that wParam was not used.

■ WM\_MDITILE (3.0)

WM\_MDITILE

fTile = wParam; /\* tiling flag \*/

The WM\_MDITILE message is sent to a multiple document interface (MDI) client window to arrange all its child windows in a tiled format.

Parameter	Description		Meaning
fTile	Value of wParam. Specifies a tiling flag. This parameter can be one of the following flags:		
	Value		
	MDITILE_HORIZONTAL		Tiles MDI child windows so that they are wide rather than tall.
	MDITILE_SKIPDISABLED		Prevents disabled MDI child windows from being tiled.
	MDITILE_VERTICAL		Tiles MDI child windows so that they are tall rather than wide.

Returns

An application should return zero if it processes this message.

See Also

WM\_MDICASCADE, WM\_MDIICONARRANGE

## Windows 3.1 changes

The following tiling flags have been added:

Value	Meaning
MDITILE_HORIZONTAL	Tiles MDI child windows horizontally (one window appears beside another).
MDITILE_SKIPDISABLED	Prevents disabled MDI child windows from being tiled.
MDITILE_VERTICAL	Tiles MDI child windows vertically (one window appears above another).

## WM\_MEASUREITEM (3.0)

```
WM_MEASUREITEM
nIDCtl = (int) wParam; /* control identifier */
lpmisCtl = (MEASUREITEMSTRUCT FAR*) lParam; /* address of structure */
```

The WM\_MEASUREITEM message is sent to the owner of an owner-drawn button, combo box, list box, or menu item when the control is created. When the owner receives the message, the owner fills in the MEASUREITEMSTRUCT structure pointed to by the lpmisCtl message parameter and returns; this informs Windows of the dimensions of the control. If a list box or combo box is created with the LBS\_OWNERDRAWVARIABLE or CBS\_OWNERDRAWVARIABLE style, this message is sent to the owner for each item in the control; otherwise, this message is sent once.

Parameter	Description
nIDCtl	Value of wParam. Specifies the identifier of the control that sent the WM_MEASUREITEM message. This parameter is 0 if the message was sent by a menu. This parameter is -1 when the system is requesting the dimensions of an edit control in an owner-drawn combo box.
lpmisCtl	Value of lParam. Points to a <u>MEASUREITEMSTRUCT</u> structure that contains the dimensions of the owner-drawn control.

### Returns

An application should return TRUE if it processes this message.

### Comments

Windows sends the WM\_MEASUREITEM message to the owner of a combo box or list box created with the OWNERDRAWFIXED style before sending WM\_INITDIALOG. As a result, when the owner receives this message, Windows has not yet determined the height and width of the font used in the control; function calls and calculations requiring these values should occur in the main function of the application or library.

### See Also

WM\_COMPAREITEM, WM\_DELETEITEM, WM\_DRAWITEM, WM\_INITDIALOG, MEASUREITEMSTRUCT

### Windows 3.1 changes

The meaning of the wParam parameter has changed. If a control sends the WM\_MEASUREITEM message, the wParam parameter specifies the identifier of the control. If a menu sends the message, wParam is zero.

## WM\_MENUCHAR (2.x)

### WM\_MENUCHAR

```
chUser = wParam; /* ASCII character */
fMenu = LOWORD(lParam); /* menu flag */
hmenu = (HMENU) HIWORD(lParam); /* handle of the menu */
```

The WM\_MENUCHAR message is sent when the user presses the key corresponding to a menu mnemonic character that doesn't match any of the predefined mnemonics in the current menu. It is sent to the window that owns the menu.

Parameter	Description						
chUser	Value of wParam. Specifies the ASCII character that corresponds to the key the user pressed.						
fMenu	Value of the low-order word of lParam. Specifies the type of the selected menu. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MF_POPUP</td><td>The menu is a pop-up menu.</td></tr><tr><td>MF_SYSMENU</td><td>The menu is a System menu (sometimes referred to as a Control menu).</td></tr></tbody></table>	Value	Meaning	MF_POPUP	The menu is a pop-up menu.	MF_SYSMENU	The menu is a System menu (sometimes referred to as a Control menu).
Value	Meaning						
MF_POPUP	The menu is a pop-up menu.						
MF_SYSMENU	The menu is a System menu (sometimes referred to as a Control menu).						
hmenu	Value of the high-order word of lParam. Identifies the selected menu.						

### Returns

The return value is one of the following command code values in the high-order word:

Value	Description
0	Informs Windows that it should discard the character corresponding to the key the user pressed, and creates a short beep on the system speaker.
1	Informs Windows that it should close the current menu.
2	Informs Windows that the low-order word of the return value contains the item number for a specific item. This item is selected by Windows.

The low-order word is ignored if the high-order word contains 0 or 1. An application should process this message when an accelerator key has been used to select a bitmap placed in a menu.

### Comments

The WM\_MENUCHAR message is generated when the user presses ALT and any key, even if the key does not correspond to a mnemonic character. In this case, the hmenu parameter contains the window handle of the menu.

## WM\_MENUSELECT (2.x)

### WM\_MENUSELECT

```
wIDItem = wParam; /* item identifier or menu handle */  
fwMenu = LOWORD(lParam); /* menu flags */  
hmenu = (HMENU) HIWORD(lParam); /* handle of the menu */
```

The WM\_MENUSELECT message is sent to the window associated with a menu when the user selects a menu item.

Parameter	Description																				
wIDItem	Value of wParam. Specifies the menu-item identifier if the selected item is a menu item. If the selected item contains a pop-up menu, wIDItem contains the handle of the pop-up menu.																				
fwMenu	Low word of lParam. Specifies one or more menu flags. This parameter can be a combination of the following values: <table><thead><tr><th>Flag</th><th>Description</th></tr></thead><tbody><tr><td>MF_BITMAP</td><td>Item is a bitmap.</td></tr><tr><td>MF_CHECKED</td><td>Item is checked.</td></tr><tr><td>MF_DISABLED</td><td>Item is disabled.</td></tr><tr><td>MF_GRAYED</td><td>Item is grayed.</td></tr><tr><td>MF_MOUSESELECT</td><td>Item was selected with a mouse.</td></tr><tr><td>MF_OWNERDRAW</td><td>Item is an owner-drawn item.</td></tr><tr><td>MF_POPUP</td><td>Item contains a pop-up menu.</td></tr><tr><td>MF_SEPARATOR</td><td>Item is a menu-item separator.</td></tr><tr><td>MF_SYSMENU</td><td>Item is contained in the System menu (sometimes referred to as the Control menu). The hmenu parameter identifies the System menu associated with the message.</td></tr></tbody></table>	Flag	Description	MF_BITMAP	Item is a bitmap.	MF_CHECKED	Item is checked.	MF_DISABLED	Item is disabled.	MF_GRAYED	Item is grayed.	MF_MOUSESELECT	Item was selected with a mouse.	MF_OWNERDRAW	Item is an owner-drawn item.	MF_POPUP	Item contains a pop-up menu.	MF_SEPARATOR	Item is a menu-item separator.	MF_SYSMENU	Item is contained in the System menu (sometimes referred to as the Control menu). The hmenu parameter identifies the System menu associated with the message.
Flag	Description																				
MF_BITMAP	Item is a bitmap.																				
MF_CHECKED	Item is checked.																				
MF_DISABLED	Item is disabled.																				
MF_GRAYED	Item is grayed.																				
MF_MOUSESELECT	Item was selected with a mouse.																				
MF_OWNERDRAW	Item is an owner-drawn item.																				
MF_POPUP	Item contains a pop-up menu.																				
MF_SEPARATOR	Item is a menu-item separator.																				
MF_SYSMENU	Item is contained in the System menu (sometimes referred to as the Control menu). The hmenu parameter identifies the System menu associated with the message.																				
hmenu	High word of lParam. If the fwMenu parameter contains the MF_SYSMENU flag, this parameter specifies the menu handle of the System menu.																				

### Returns

An application should return zero if it processes this message.

### Comments

If the fwMenu parameter contains -1 and the hmenu parameter contains 0, Windows has closed the menu. This occurs both when the menu is closed because the user pressed ESC or clicked outside the menu and when the user has selected a menu item.



## WM\_MOUSEACTIVATE (2.x)

### WM\_MOUSEACTIVATE

```
hwndTopLevel = (HWND) wParam; /* handle of top-level parent */
wHitTestCode = LOWORD(lParam); /* hit-test code */
wMsg = HIWORD(lParam); /* mouse-message identifier */
```

The WM\_MOUSEACTIVATE message is sent when the cursor is in an inactive window and the user presses a mouse button. The parent window receives this message only if the child window passes it to the DefWindowProc function.

Parameter	Description
hwndTopLevel	Value of wParam. Identifies the top-level parent window of the window being activated.
wHitTestCode	Value of the low-order word of lParam. Specifies the hit-test area code. A hit test is a test that determines the location of the cursor.
wMsg	Value of the high-order word of lParam. Specifies the identifier of the mouse message.

### Returns

The return value specifies whether the window should be activated and whether the mouse event should be discarded. It must be one of the following values:

Value	Meaning
MA_ACTIVATE	Activate the window.
MA_NOACTIVATE	Do not activate the window.
MA_ACTIVATEANDEAT	Activate the window and discard the mouse event.
MA_NOACTIVATEANDEAT	Do not activate the window; discard the mouse event.

### Comments

If the child window passes the message to the DefWindowProc function, DefWindowProc passes this message to a window's parent window before any processing occurs. If the parent window returns a nonzero value, processing is halted.

### See Also

DefWindowProc

## Windows 3.1 changes

The following returns were added:

Value

MA\_ACTIVATE

MA\_NOACTIVATE

MA\_ACTIVATEANDEAT

MA\_NOACTIVATEANDEAT

Meaning

Activate the window.

Do not activate the window.

Activate the window and discard the mouse event.

Do not activate the window and discard the mouse event.

## WM\_MOUSEMOVE (2.x)

### WM\_MOUSEMOVE

```
fwKeys = wParam; /* key flags */
xPos = LOWORD(lParam); /* horizontal position of cursor */
yPos = HIWORD(lParam); /* vertical position of cursor */
```

The WM\_MOUSEMOVE message is sent to a window when the mouse cursor moves. If the mouse is not captured, the message goes to the window beneath the cursor. Otherwise, the message goes to the window that has captured the mouse.

Parameter	Description												
fwKeys	Value of wParam. Indicates whether various virtual keys are down. This parameter can be any combination of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>MK_CONTROL</td><td>Set if CTRL key is down.</td></tr><tr><td>MK_LBUTTON</td><td>Set if left button is down.</td></tr><tr><td>MK_MBUTTON</td><td>Set if middle button is down.</td></tr><tr><td>MK_RBUTTON</td><td>Set if right button is down.</td></tr><tr><td>MK_SHIFT</td><td>Set if SHIFT key is down.</td></tr></tbody></table>	Value	Description	MK_CONTROL	Set if CTRL key is down.	MK_LBUTTON	Set if left button is down.	MK_MBUTTON	Set if middle button is down.	MK_RBUTTON	Set if right button is down.	MK_SHIFT	Set if SHIFT key is down.
Value	Description												
MK_CONTROL	Set if CTRL key is down.												
MK_LBUTTON	Set if left button is down.												
MK_MBUTTON	Set if middle button is down.												
MK_RBUTTON	Set if right button is down.												
MK_SHIFT	Set if SHIFT key is down.												
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor, as a screen coordinate.												
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor, as a screen coordinate.												

### Returns

An application should return zero if it processes this message.

### Comments

The MAKEPOINT macro can be used to convert the lParam parameter to a POINT structure.

### See Also

SetCapture, WM\_NCHITTEST, MAKEPOINT, POINT

## WM\_MOVE (2.x)

WM\_MOVE

```
xPos = (int) LOWORD(lParam);    /* horizontal position */  
yPos = (int) HIWORD(lParam);    /* vertical position   */
```

The WM\_MOVE message is sent after a window has been moved.

Parameter	Description
xPos	Value of the low-order word of lParam. Specifies the new x-coordinate of the upper-left corner of the client area of the window.
yPos	Value of the high-order word of lParam. Specifies the new y-coordinate of the upper-left corner of the client area of the window.

### Returns

An application should return zero if it processes this message.

### Comments

The xPos and yPos parameters are given in screen coordinates for overlapped and pop-up windows and in parent-client coordinates for child windows.

An application can use the MAKEPOINT macro to convert the lParam parameter to a POINT data structure.

### See Also

MAKEPOINT, POINT

## WM\_NCACTIVATE (2.x)

WM\_NCACTIVATE

```
fActive = (BOOL) wParam; /* the active/inactive flag */
```

The WM\_NCACTIVATE message is sent to a window when its nonclient area needs to be changed to indicate an active or inactive state.

Parameter	Description
fActive	Value of wParam. Specifies when a title bar or icon needs to be changed to indicate an active or inactive state. The fActive parameter is TRUE if an active title bar or icon is to be drawn. It is FALSE for an inactive title bar or icon.

### Returns

When the fActive parameter is FALSE, an application should return TRUE to indicate that Windows should proceed with the default processing or FALSE to prevent the caption bar or icon from being deactivated. When fActive is TRUE, the return value is ignored.

### Comments

The DefWindowProc function draws the title bar and title bar text in their active colors when the fActive parameter is TRUE and in their inactive colors when fActive is FALSE.

### See Also

DefWindowProc

## WM\_NCCALCSIZE (2.x)

```
WM_NCCALCSIZE
fCalcValidRects = (BOOL) wParam; /* valid-area flag */
lpncsp = (NCCALCSIZE_PARAMS FAR*) lParam; /* address of data */
```

The WM\_NCCALCSIZE message is sent when the size and position of a window's client area needs to be calculated. By processing this message, an application can control the contents of the window's client area when the size or position of the window changes.

Parameter	Description
fCalcValidRects	Value of wParam. Specifies whether the application should specify which part of the client area contains valid information. Windows will copy the valid information to the specified area within the new client area. If this parameter is TRUE, the application should specify which part of the client area is valid.
lpncsp	Value of lParam. Points to an NCCALCSIZE_PARAMS data structure that contains information an application can use to calculate the new size and position of the client rectangle.  Regardless of the value of fCalcValidRects, the first rectangle in the array specified by the rgrc member contains the coordinates of the window. For a child window, the coordinates are relative to the parent window's client area. For top-level windows, the coordinates are screen coordinates. An application should process WM_NCCALCSIZE by modifying the rgrc[0] rectangle to reflect the size and position of the client area.  The rgrc[1] and rgrc[2] rectangles are valid only if fCalcValidRects is TRUE. In this case, the rgrc[1] rectangle contains the coordinates of the window before it was moved or resized. The rgrc[2] rectangle contains the coordinates of the window's client area before the window was moved. All coordinates are relative to the parent window or screen.

### Returns

An application should return zero if fCalcValidRects is FALSE.

An application can return zero or a valid combination of the following values if fCalcValidRects is TRUE:

#### Value Meaning

WVR\_ALIGNTOP, WVR\_ALIGNLEFT, WVR\_ALIGNBOTTOM, WVR\_ALIGNRIGHT

These values, used in combination, specify that the client area of the window is to be preserved and aligned appropriately relative to the new location of the client window. For example, to align the client area to the lower-left, return WVR\_ALIGNLEFT | WVR\_ALIGNTOP.

WVR\_HREDRAW, WVR\_VREDRAW

These values, used in combination with any other values, cause the window to be completely redrawn if the client rectangle changed size horizontally or vertically. These values are similar to the CS\_HREDRAW and CS\_VREDRAW class styles.

WVR\_REDRAW

This value causes the entire window to be redrawn. It is a combination of WVR\_HREDRAW and WVR\_VREDRAW.

WVR\_VALIDRECTS

This value indicates that, upon return from WM\_NCCALCSIZE, the rgrc[1] and rgrc[2] rectangles contain valid source and destination area rectangles, respectively. Windows combines these rectangles to calculate the area of the window that can be preserved. Windows copies any part of the window image that is within the source rectangle and clips the image to the destination rectangle. Both rectangles are in parent-relative or screen-relative coordinates.

This return value allows an application to implement more elaborate client-area preservation strategies, such as centering or preserving a subset of the client area.

If fCalcValidRects is TRUE and an application returns zero, the old client area is preserved and is aligned with the upper-left corner of the new client area.

### Comments

Redrawing of the window may occur, depending on whether CS\_HREDRAW or CS\_VREDRAW was specified. This is the default, backward-compatible DefWindowProc processing of this message (in addition to the usual client rectangle calculation described in the preceding table).

See Also

DefWindowProc, MoveWindow, SetWindowPos, NCCALCSIZE\_PARAMS, RECT

## WM\_NCCREATE (2.x)

WM\_NCCREATE

```
lpcs = (CREATESTRUCT FAR*) lParam; /* address of initialization data *  
/
```

The WM\_NCCREATE message is sent prior to the WM\_CREATE message when a window is first created.

Parameter	Description
lpcs	Value of lParam. Points to the <u>CREATESTRUCT</u> data structure for the window.

### Returns

The return value is nonzero if the nonclient area is created. It is zero if an error occurs; in this case, the CreateWindow or CreateWindowEx function will return NULL.

### Comments

Scroll bars are initialized (the scroll bar position and range are set), and the window text is set. Memory used internally to create and maintain the window is allocated.

### See Also

CreateWindow, WM\_CREATE, CREATESTRUCT



## WM\_NCDESTROY (2.x)

### WM\_NCDESTROY

The WM\_NCDESTROY message informs a window that its nonclient area is being destroyed. The DestroyWindow function sends the WM\_NCDESTROY message to the window following the WM\_DESTROY message. WM\_NCDESTROY is used to free the allocated memory object associated with the window.

#### Parameters

This message has no parameters.

#### Returns

An application should return zero if it processes this message.

#### Comments

This message frees any memory internally allocated for the window.

#### See Also

DestroyWindow, WM\_NCCREATE

## WM\_NCHITTEST (2.x)

```
WM_NCHITTEST
xPos = (int) LOWORD(lParam);    /* horizontal position of cursor */
yPos = (int) HIWORD(lParam);    /* vertical position of cursor  */
```

The WM\_NCHITTEST message is sent to the window that contains the cursor or to the window that used the SetCapture function to capture the mouse input. It is sent every time the mouse is moved.

Parameter	Description
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor, in screen coordinates.
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor, in screen coordinates.

### Returns

The return value of the DefWindowProc function is one of the following values indicating the position of the cursor:

Value	Meaning
HTBORDER	In the border of a window that does not have a sizing border
HTBOTTOM	In the lower horizontal border of a window
HTBOTTOMLEFT	In the lower-left corner of a window border
HTBOTTOMRIGHT	In the lower-right corner of a window border
HTCAPTION	In a title bar area
HTCLIENT	In a client area
HTERROR	On the screen background or on a dividing line between windows (same as HTNOWHERE except that the <u>DefWindowProc</u> function produces a system beep to indicate an error)
HTGROWBOX	In a size box (same as HTSIZE)
HTHSCROLL	In the horizontal scroll bar
HTLEFT	In the left border of a window
HTMAXBUTTON	In a Maximize button
HTMENU	In a menu area
HTMINBUTTON	In a Minimize button
HTNOWHERE	On the screen background or on a dividing line between windows
HTREDUCE	In a Minimize button
HTRIGHT	In the right border of a window
HTSIZE	In a size box (same as HTGROWBOX)
HTSYSMENU	In a System menu (sometimes referred to as a Control menu) or in a close button in a child window
HTTOP	In the upper horizontal border of a window
HTTOPLEFT	In the upper-left corner of a window border
HTTOPRIGHT	In the upper-right corner of a window border
HTTRANSPARENT	In a window currently covered by another window
HTVSCROLL	In the vertical scroll bar
HTZOOM	In a Maximize button

### Comments

The MAKEPOINT macro can be used to convert the lParam parameter to a POINT structure.

### Example

This example shows a portion of a subclass procedure that detects mouse messages in a static window:

```
LONG lRetVal;

case WM_NCHITTEST:
    lRetVal = DefWindowProc(hwnd, msg, wParam, lParam);
    if (lRetVal == HTTRANSPARENT) {
```

```
. /* Process mouse events in static window. */  
. }  
break;
```

default:

```
CallWindowProc(lpStaticProc, hwnd, msg, wParam, lParam);
```

See Also

DefWindowProc, GetCapture, MAKEPOINT, POINT

## Windows 3.1 changes

The following hit-test code was previously undocumented:

Value	Meaning
HTBORDER	In the border of a window that does not have a sizing border.

## WM\_NCLBUTTONDBLCLK (2.x)

```
WM_NCLBUTTONDBLCLK
nHittest = wParam; /* hit-test code */
xCursor = LOWORD(lParam); /* cursor horizontal position */
yCursor = HIWORD(lParam); /* cursor vertical position */
```

The WM\_NCLBUTTONDBLCLK message is sent when the user double-clicks the left mouse button while the cursor is within a nonclient area of the window.

Parameter	Description
nHittest	Value of wParam. Specifies the code returned by WM_NCHITTEST. For more information, see the description of the <u>WM_NCHITTEST</u> message.
xCursor	Value of the low-order word of lParam. Specifies the horizontal position of the cursor, in screen coordinates.
yCursor	Value of the high-order word of lParam. Specifies the vertical position of the cursor, in screen coordinates.

### Returns

An application should return zero if it processes this message.

### Comments

If appropriate, WM\_SYSCOMMAND messages are sent.

### See Also

WM\_NCHITTEST, WM\_SYSCOMMAND, POINT

## WM\_NCLBUTTONDOWN (2.x)

WM\_NCLBUTTONDOWN

```
wHitTestCode = wParam;    /* hit-test code    */
xPos = LOWORD(lParam);    /* horizontal cursor position */
yPos = HIWORD(lParam);    /* vertical cursor position  */
```

The WM\_NCLBUTTONDOWN message is sent to a window when the user presses the left mouse button while the cursor is within a nonclient area of the window.

Parameter	Description
wHitTestCode	Value of wParam. Specifies the code returned by WM_NCHITTEST. For more information, see the description of the <u>WM_NCHITTEST</u> message.
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor, in screen coordinates.
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor, in screen coordinates.

### Returns

An application should return zero if it processes this message.

### Comments

If appropriate, WM\_SYSCOMMAND messages are sent.

### See Also

WM\_NCHITTEST, WM\_NCLBUTTONDBLCLK, WM\_NCLBUTTONUP, WM\_SYSCOMMAND, POINT

## WM\_NCLBUTTONUP (2.x)

### WM\_NCLBUTTONUP

```
wHitTestCode = wParam; /* hit-test code */
xPos = LOWORD(lParam); /* horizontal cursor position */
yPos = HIWORD(lParam); /* vertical cursor position */
```

The WM\_NCLBUTTONUP message is sent to a window when the user releases the left mouse button while the cursor is within a nonclient area of the window.

Parameter	Description
wHitTestCode	Value of wParam. Specifies the code returned by WM_NCHITTEST. For more information, see the description of the <u>WM_NCHITTEST</u> message.
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor, in screen coordinates.
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor, in screen coordinates.

### Returns

An application should return zero if it processes this message.

### Comments

If appropriate, WM\_SYSCOMMAND messages are sent.

### See Also

WM\_NCHITTEST, WM\_NCLBUTTONDOWN, WM\_NCLBUTTONUP, WM\_SYSCOMMAND

## WM\_NCMBUTTONDBLCLK (2.x)

```
WM_NCMBUTTONDBLCLK
wHitTestCode = wParam;    /* hit-test code    */
xPos = LOWORD(lParam);    /* horizontal cursor position */
yPos = HIWORD(lParam);    /* vertical cursor position  */
```

The WM\_NCRBUTTONDOWN message is sent to a window when the user double-clicks the middle mouse button while the cursor is within a nonclient area of the window.

Parameter	Description
wHitTestCode	Value of wParam. Specifies the code returned by WM_NCHITTEST. For more information, see the description of the <u>WM_NCHITTEST</u> message.
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor, as a screen coordinate.
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor, as a screen coordinate.

### Returns

An application should return zero if it processes this message.

### See Also

WM\_NCHITTEST, WM\_NCMBUTTONDOWN, WM\_NCMBUTTONUP, POINT



## WM\_NCMBUTTONDOWN (2.x)

### WM\_NCMBUTTONDOWN

```
wHitTestCode = wParam;    /* hit-test code    */  
xPos = LOWORD(lParam);    /* horizontal cursor position */  
yPos = HIWORD(lParam);    /* vertical cursor position  */
```

The WM\_NCMBUTTONDOWN message is sent to a window when the user presses the middle mouse button while the cursor is within a nonclient area of the window.

Parameter	Description
wHitTestCode	Value of wParam. Specifies the code returned by WM_NCHITTEST. For more information, see the description of the <u>WM_NCHITTEST</u> message.
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor, as a screen coordinate.
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor, as a screen coordinate.

### Returns

An application should return zero if it processes this message.

### See Also

WM\_NCHITTEST, WM\_NCMBUTTONDBLCLK, WM\_NCMBUTTONUP

## WM\_NCMBUTTONUP (2.x)

### WM\_NCMBUTTONUP

```
wHitTestCode = wParam; /* hit-test code */
xPos = LOWORD(lParam); /* horizontal cursor position */
yPos = HIWORD(lParam); /* vertical cursor position */
```

The WM\_NCMBUTTONUP message is sent to a window when the user releases the left mouse button while the cursor is within a nonclient area of the window.

Parameter	Description
wHitTestCode	Value of wParam. Specifies the code returned by WM_NCHITTEST. For more information, see the description of the <u>WM_NCHITTEST</u> message.
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor, as a screen coordinate.
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor, as a screen coordinate.

### Returns

An application should return zero if it processes this message.

### See Also

WM\_NCHITTEST, WM\_NCMBUTTONDBLCLK, WM\_NCMBUTTONDOWN

## WM\_NCMOUSEMOVE (2.x)

```
WM_NCMOUSEMOVE
wHitTestCode = wParam;    /* hit-test code    */
xPos = LOWORD(lParam);    /* horizontal cursor position */
yPos = HIWORD(lParam);    /* vertical cursor position  */
```

The WM\_NCMOUSEMOVE message is sent to a window when the cursor is moved within a nonclient area of the window.

Parameter	Description
wHitTestCode	Value of wParam. Specifies the code returned by WM_NCHITTEST. For more information, see the description of the <u>WM_NCHITTEST</u> message.
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor, as a screen coordinate.
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor, as a screen coordinate.

### Returns

An application should return zero if it processes this message.

### Comments

If appropriate, WM\_SYSCOMMAND messages are sent.

### See Also

WM\_NCHITTEST, WM\_SYSCOMMAND, POINT

## WM\_NCPAINT (2.x)

### WM\_NCPAINT

The WM\_NCPAINT message is sent to a window when its frame needs painting.

#### Parameters

This message has no parameters.

#### Returns

An application should return zero if it processes this message.

#### Comments

The DefWindowProc function paints the window frame.

An application can intercept this message and paint its own custom window frame. The clipping region for a window is always rectangular, even if the shape of the frame is altered.

#### See Also

DefWindowProc

## WM\_NCRBUTTONDBLCLK (2.x)

```
WM_NCRBUTTONDBLCLK
wHitTestCode = wParam;    /* hit-test code    */
xPos = LOWORD(lParam);    /* horizontal cursor position */
yPos = HIWORD(lParam);    /* vertical cursor position  */
```

The WM\_NCRBUTTONDBLCLK message is sent to a window when the user double-clicks the right mouse button while the cursor is within a nonclient area of the window.

Parameter	Description
wHitTestCode	Value of wParam. Specifies the code returned by WM_NCHITTEST. For more information, see the description of the <u>WM_NCHITTEST</u> message.
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor, as a screen coordinate.
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor, as a screen coordinate.

### Returns

An application should return zero if it processes this message.

### See Also

WM\_NCHITTEST, WM\_NCRBUTTONDOWN, WM\_NCRBUTTONUP, POINT

## WM\_NCRBUTTONDOWN (2.x)

### WM\_NCRBUTTONDOWN

```
wHitTestCode = wParam;    /* hit-test code    */  
xPos = LOWORD(lParam);    /* horizontal cursor position */  
yPos = HIWORD(lParam);    /* vertical cursor position  */
```

The WM\_NCRBUTTONDOWN message is sent to a window when the user presses the right mouse button while the cursor is within a nonclient area of the window.

Parameter	Description
wHitTestCode	Value of wParam. Specifies the code returned by WM_NCHITTEST. For more information, see the description of the <u>WM_NCHITTEST</u> message.
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor, as a screen coordinate.
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor, as a screen coordinate.

### Returns

An application should return zero if it processes this message.

### See Also

WM\_NCHITTEST, WM\_NCRBUTTONDBLCLK, WM\_NCRBUTTONUP, POINT

## WM\_NCRBUTTONUP (2.x)

```
WM_NCRBUTTONUP
wHitTestCode = wParam;    /* hit-test code    */
xPos = LOWORD(lParam);    /* horizontal cursor position */
yPos = HIWORD(lParam);    /* vertical cursor position  */
```

The WM\_NCRBUTTONUP message is sent to a window when the user releases the right mouse button while the cursor is within a nonclient area of the window.

Parameter	Description
wHitTestCode	Value of wParam. Specifies the code returned by WM_NCHITTEST. For more information, see the description of the <u>WM_NCHITTEST</u> message.
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor, as a screen coordinate.
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor, as a screen coordinate.

### Returns

An application should return zero if it processes this message.

### See Also

WM\_NCHITTEST, WM\_NCRBUTTONDBLCLK, WM\_NCRBUTTONDOWN, POINT

## WM\_NEXTDLGCTL (2.x)

WM\_NEXTDLGCTL

```
wCtlFocus = wParam; /* identifies control for focus */  
fHandle = (BOOL) LOWORD(lParam); /* wParam handle flag */
```

An application sends the WM\_NEXTDLGCTL message to a dialog box procedure to set the focus to a different control in a dialog box.

Parameter	Description
wCtlFocus	Value of wParam. If the fHandle parameter is nonzero, the wCtlFocus parameter is the handle of the control that receives the focus. If fHandle is zero, wCtlFocus is a flag that indicates whether the next or previous control with the <u>WS_TABSTOP</u> style receives the focus. If wCtlFocus is zero, the next control receives the focus; otherwise, the previous control with the <u>WS_TABSTOP</u> style receives the focus.
fHandle	Low-order word of lParam. Indicates how Windows uses the wParam parameter. If fHandle is nonzero, wParam is a handle associated with the control that receives the focus; otherwise, wParam is a flag that indicates whether the next or previous control with the <u>WS_TABSTOP</u> style receives the focus.

### Returns

An application should return zero if it processes this message.

### Comments

The effect of this message differs from that of the SetFocus function because WM\_NEXTDLGCTL modifies the border around the default button.

Do not use the SendMessage function to send a WM\_NEXTDLGCTL message if your application will concurrently process other messages that set the control focus. In this case, use the PostMessage function instead.

### See Also

PostMessage, SendMessage, SetFocus



## WM\_PAINT (2.x)

### WM\_PAINT

The WM\_PAINT message is sent when Windows or an application makes a request to repaint a portion of an application's window. The message is sent when the UpdateWindow or RedrawWindow function is called or by the DispatchMessage function when the application obtains a WM\_PAINT message by using the GetMessage or PeekMessage function.

#### Parameters

This message has no parameters.

#### Returns

An application should return zero if it processes this message.

#### Comments

The DispatchMessage function sends this message when there are no other messages in the application's message queue.

A window may receive internal paint messages as a result of calling the RedrawWindow function with the RDW\_INTERNALPAINT flag set. In this case, the window may not have an update region. An application should call the GetUpdateRect function to determine whether the window has an update region. If GetUpdateRect returns zero, the application should not call the BeginPaint and EndPaint functions.

It is an application's responsibility to check for any necessary internal repainting or updating by looking at its internal data structures for each WM\_PAINT message, because a WM\_PAINT message may have been caused by both an invalid area and a call to the RedrawWindow function with the RDW\_INTERNALPAINT flag set.

An internal WM\_PAINT message is sent only once by Windows. After an internal WM\_PAINT message is returned from the GetMessage or PeekMessage function or is sent to a window by the UpdateWindow function, no further WM\_PAINT messages will be sent or posted until the window is invalidated or until the RedrawWindow function is called again with the RDW\_INTERNALPAINT flag set.

#### See Also

BeginPaint, DispatchMessage, EndPaint, GetMessage, PeekMessage, RedrawWindow, UpdateWindow

### Windows 3.1 changes

A window that may receive internal paint messages as a result of calling the RedrawWindow function with the RDW\_INTERNALPAINT flag set. In this case, the window may not have an update region. An application should call the GetUpdateRect function to determine whether the window has an update region. If GetUpdateRect returns zero, the application should not call the BeginPaint and EndPaint functions.

It is an application's responsibility to check for any necessary internal repainting or updating by looking at its internal data structures for each WM\_PAINT message, since a WM\_PAINT message may have been caused by both an invalid area and a call to the RedrawWindow function with the RDW\_INTERNALPAINT flag set.

Internal WM\_PAINT messages are only sent once by Windows. After an internal WM\_PAINT message is returned from the GetMessage or PeekMessage function, or sent to a window by the UpdateWindow function, no further WM\_PAINT messages will be sent or posted until the window is invalidated or until the RedrawWindow function is called again with the RDW\_INTERNALPAINT flag set.

## WM\_PAINTCLIPBOARD (2.x)

```
WM_PAINTCLIPBOARD
hwndViewer = (HWND) wParam; /* handle of viewer*/
pps = (PAINTSTRUCT FAR*) LOWORD(lParam); /* points to paint data */
```

The WM\_PAINTCLIPBOARD message is sent by a clipboard viewer to the clipboard owner when the owner has placed data on the clipboard in the CF\_OWNERDISPLAY format and the clipboard viewer's client area needs repainting.

Parameter	Description
hwndViewer	Value of wParam. Specifies a handle to the clipboard viewer window.
pps	Value of the low-order word of lParam. Points to a <u>PAINTSTRUCT</u> data structure that defines which part of the client area to paint.

### Returns

An application should return zero if it processes this message.

### Comments

To determine whether the entire client area or just a portion of it needs repainting, the clipboard owner must compare the dimensions of the drawing area given in the rcPaint member of the PAINTSTRUCT structure to the dimensions given in the most recent WM\_SIZECLIPBOARD message.

An application must use the GlobalLock function to lock the memory that contains the PAINTSTRUCT data structure. The application should unlock that memory by using the GlobalUnlock function before it yields or returns control.

### See Also

GlobalLock, GlobalUnlock, WM\_SIZECLIPBOARD, PAINTSTRUCT

## WM\_PALETTECHANGED (3.0)

WM\_PALETTECHANGED

```
hwndPalChg = (HWND) wParam; /* handle of window that changed palette */
```

The WM\_PALETTECHANGED message is sent to all top-level and overlapped windows after the window with the input focus has realized its logical palette, thereby changing the system palette. This message allows a window without the input focus that uses a color palette to realize its logical palette and update its client area.

Parameter	Description
hwndPalChg	Value of wParam. Specifies the handle of the window that caused the system palette to change.

### Returns

An application should return zero if it processes this message.

### Comments

This message is sent to all top-level and overlapped windows, including the one that changed the system palette and caused this message to be sent. If any child windows use a color palette, this message must be passed on to them.

To avoid an infinite loop, a window that receives this message should not realize its palette unless it determines that wParam does not contain its own window handle.

### Example

This example shows how an application selects and realizes its logical palette:

```
HDC hdc;
HPALETTE hpalApp, hpalT;
UINT i;

/*
 * If this application changed the palette, ignore the message.
 */

case WM_PALETTECHANGED:
    if ((HWND) wParam == hwnd)
        return 0;

/* Otherwise, fall through to WM_QUERYNEWPALETTE. */

case WM_QUERYNEWPALETTE:

    /*
    * If realizing the palette causes the palette to change,
    * redraw completely.
    */

    hdc = GetDC(hwnd);
    hpalT = SelectPalette (hdc, hpalApp, FALSE);

    i = RealizePalette(hdc); /* i == entries that changed */

    SelectPalette (hdc, hpalT, FALSE);
    ReleaseDC(hwnd, hdc);

    /* If any palette entries changed, repaint the window. */

    if (i > 0)
        InvalidateRect(hwnd, NULL, TRUE);

    return i;
```

See Also

WM\_PALETTEISCHANGING, WM\_QUERYNEWPALETTE, RealizePalette

## WM\_PALETTEISCHANGING (3.1)

WM\_PALETTEISCHANGING

```
hwndRealize = (HWND) wParam; /* handle of window to realize palette */
```

The WM\_PALETTEISCHANGING message informs applications that an application is going to realize its logical palette.

Parameter	Description
hwndRealize	Value of wParam. Specifies the handle of the window that is going to realize its logical palette.

### Returns

An application should return zero if it processes this message.

### See Also

WM\_PALETTECHANGED, WM\_QUERYNEWPALETTE

## WM\_PARENTNOTIFY (3.0)

```
WM_PARENTNOTIFY
fwEvent = wParam;      /* event flags */
wValue1 = LOWORD(lParam); /* child handle/cursor x-coordinate */
wValue2 = HIWORD(lParam); /* child ID/cursor y-coordinate*/
```

The WM\_PARENTNOTIFY message is sent to the parent of a child window when the child window is created or destroyed or when the user clicks a mouse button while the cursor is over the child window. When the child window is being created, the system sends WM\_PARENTNOTIFY just before the CreateWindow or CreateWindowEx function that creates the window returns. When the child window is being destroyed, the system sends the message before any processing to destroy the window takes place.

Parameter	Description												
fwEvent	Value of wParam. Specifies the event for which the parent is being notified. It can be any of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>WM_CREATE</td><td>The child window is being created.</td></tr><tr><td>WM_DESTROY</td><td>The child window is being destroyed.</td></tr><tr><td>WM_LBUTTONDOWN</td><td>The user has placed the mouse cursor over the child window and clicked the left mouse button.</td></tr><tr><td>WM_MBUTTONDOWN</td><td>The user has placed the mouse cursor over the child window and clicked the middle mouse button.</td></tr><tr><td>WM_RBUTTONDOWN</td><td>The user has placed the mouse cursor over the child window and clicked the right mouse button.</td></tr></tbody></table>	Value	Description	WM_CREATE	The child window is being created.	WM_DESTROY	The child window is being destroyed.	WM_LBUTTONDOWN	The user has placed the mouse cursor over the child window and clicked the left mouse button.	WM_MBUTTONDOWN	The user has placed the mouse cursor over the child window and clicked the middle mouse button.	WM_RBUTTONDOWN	The user has placed the mouse cursor over the child window and clicked the right mouse button.
Value	Description												
WM_CREATE	The child window is being created.												
WM_DESTROY	The child window is being destroyed.												
WM_LBUTTONDOWN	The user has placed the mouse cursor over the child window and clicked the left mouse button.												
WM_MBUTTONDOWN	The user has placed the mouse cursor over the child window and clicked the middle mouse button.												
WM_RBUTTONDOWN	The user has placed the mouse cursor over the child window and clicked the right mouse button.												
wValue1	Value of the low-order word of lParam. If the fwEvent parameter is <u>WM_CREATE</u> or <u>WM_DESTROY</u> , the wValue1 parameter specifies the handle of the <u>child window</u> . Otherwise, wValue1 specifies the x-coordinate of the cursor.												
wValue2	Value of the high-order word of lParam. If fwEvent is <u>WM_CREATE</u> or <u>WM_DESTROY</u> , the wValue2 parameter specifies the <u>identifier of the child window</u> . Otherwise, wValue2 specifies the y-coordinate of the cursor.												

### Returns

An application should return zero if it processes this message.

### Comments

This message is also sent to all ancestor windows of the child window, including the top-level window.

All child windows except those that have the WS\_EX\_NOPARENTNOTIFY send this message to their parent windows. By default, child windows in a dialog box have the WS\_EX\_NOPARENTNOTIFY style unless the CreateWindowEx function was called to create the child window without this style.

### See Also

CreateWindow, CreateWindowEx, WM\_CREATE, WM\_DESTROY, WM\_LBUTTONDOWN, WM\_MBUTTONDOWN, WM\_RBUTTONDOWN

## WM\_PASTE (2.x)

```
WM_PASTE  
wParam = 0; /* not used, must be zero */  
lParam = 0L; /* not used, must be zero */
```

An application sends the WM\_PASTE message to an edit control or combo box to insert the data from the clipboard into the edit control at the current cursor position. Data is inserted only if the clipboard contains data in CF\_TEXT format.

### Parameters

This message has no parameters.

### Returns

The return value is nonzero if this message is sent to an edit control or a combo box.

### Example

This example pastes data from the clipboard to an edit control:

```
SendMessage(hdlg, IDD_MYEDITCONTROL, WM_PASTE, 0, 0L);
```

### See Also

WM\_CLEAR, WM\_COPY, WM\_CUT



## WM\_POWER (3.1)

```
WM_POWER
fwPowerEvt = wParam;    /* power-event notification message */
```

The WM\_POWER message is sent when the system, typically a battery-powered personal computer, is about to enter the suspended mode.

Parameter	Description								
fwPowerEvt	Value of wParam. Specifies a power-event notification message. This parameter may be one of the following values:								
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>PWR_SUSPENDREQUEST</td><td>Indicates that the system is about to enter the suspended mode.</td></tr><tr><td>PWR_SUSPENDRESUME</td><td>Indicates that the system is resuming operation after entering the suspended mode normally--that is, the system sent a PWR_SUSPENDREQUEST notification message to the application before the system was suspended. An application should perform any necessary recovery actions.</td></tr><tr><td>PWR_CRITICALRESUME</td><td>Indicates that the system is resuming operation after entering the suspended mode without first sending a PWR_SUSPENDREQUEST notification message to the application. An application should perform any necessary recovery actions.</td></tr></tbody></table>	Value	Meaning	PWR_SUSPENDREQUEST	Indicates that the system is about to enter the suspended mode.	PWR_SUSPENDRESUME	Indicates that the system is resuming operation after entering the suspended mode normally--that is, the system sent a PWR_SUSPENDREQUEST notification message to the application before the system was suspended. An application should perform any necessary recovery actions.	PWR_CRITICALRESUME	Indicates that the system is resuming operation after entering the suspended mode without first sending a PWR_SUSPENDREQUEST notification message to the application. An application should perform any necessary recovery actions.
Value	Meaning								
PWR_SUSPENDREQUEST	Indicates that the system is about to enter the suspended mode.								
PWR_SUSPENDRESUME	Indicates that the system is resuming operation after entering the suspended mode normally--that is, the system sent a PWR_SUSPENDREQUEST notification message to the application before the system was suspended. An application should perform any necessary recovery actions.								
PWR_CRITICALRESUME	Indicates that the system is resuming operation after entering the suspended mode without first sending a PWR_SUSPENDREQUEST notification message to the application. An application should perform any necessary recovery actions.								

### Returns

The value an application should return depends on the value of the wParam parameter, as follows:

Value of wParam	Return Value
PWR_SUSPENDREQUEST	PWR_FAIL to prevent the system from entering the suspended state; otherwise PWR_OK
PWR_SUSPENDRESUME	0
PWR_CRITICALRESUME	0

### Comments

This message is sent only to an application that is running on a system that conforms to the advanced power management (APM) basic input-and-output system (BIOS) specification. The message is sent by the power-management driver to each window returned by the [EnumWindows](#) function.

The suspended mode is the state in which the greatest amount of power savings occurs, but all operational data and parameters are preserved. Random-access memory (RAM) contents are preserved, but many devices are likely to be turned off.

### See Also

[EnumWindows](#)

## WM\_QUERYDRAGICON (3.0)

### WM\_QUERYDRAGICON

The WM\_QUERYDRAGICON message is sent to a minimized (iconic) window that does not have an icon defined for its class. The system sends this message whenever it needs to display an icon for the window.

#### Parameters

This message has no parameters.

#### Returns

An application should return a doubleword value that contains a cursor or icon handle in the low-order word. The cursor or icon must be compatible with the display driver's resolution. If the application returns NULL, the system displays the default cursor. The default return value is NULL.

#### Comments

If an application returns the handle of an icon or cursor, the system converts it to black-and-white.

The application can call the LoadCursor or LoadIcon function to load a cursor or icon from the resources in its executable file and to obtain this handle.

#### Example

This example returns an icon handle in response to the WM\_QUERYDRAGICON message. The icon is loaded from the resources in the application's executable file.

```
static HICON hIcon;

switch(msg) {
    case WM_CREATE:

        /* Load icon resource. */

        hIcon = LoadIcon(hInstance, (LPCSTR) "MyIcon");
        .
        . /* Initialize other variables. */
        .

        return 0L;

    case WM_QUERYDRAGICON:

        /* Icon is about to be dragged. Return handle to custom icon. */

        return (hIcon);

        .
        . /* Process other messages. */
        .
}

```

#### See Also

LoadCursor, LoadIcon

## WM\_QUERYENDSESSION (2.x)

### WM\_QUERYENDSESSION

The WM\_QUERYENDSESSION message is sent when the user chooses to end the Windows session, or when an application calls the ExitWindows function. If any application returns zero, the Windows session is not ended. Windows stops sending WM\_QUERYENDSESSION messages as soon as one application returns zero and sends WM\_ENDSESSION messages, with the wParam parameter set to FALSE, to any applications that have already returned nonzero.

#### Parameters

This message has no parameters.

#### Returns

An application should return nonzero if it can conveniently terminate; otherwise, it should return zero.

#### Comments

The DefWindowProc function returns nonzero when it processes this message.

#### See Also

DefWindowProc, ExitWindows, WM\_ENDSESSION

## WM\_QUERYNEWPALETTE (3.0)

### WM\_QUERYNEWPALETTE

The WM\_QUERYNEWPALETTE message informs an application that it is about to receive the input focus, giving the application an opportunity to realize its logical palette when it receives the focus.

#### Parameters

This message has no parameters.

#### Returns

An application should return nonzero if it realizes its logical palette; otherwise, it should return zero.

#### Example

This example shows how an application selects and realizes its logical palette:

```
HDC hdc;
HPALETTE hpalApp, hpalT;
UINT i;

/*
 * If this application changed the palette, ignore the message.
 */

case WM_PALETTECHANGED:
    if ((HWND) wParam == hwnd)
        return 0;

/* Otherwise, fall through to WM_QUERYNEWPALETTE. */

case WM_QUERYNEWPALETTE:

    /*
    * If realizing the palette causes the palette to change,
    * redraw completely.
    */

    hdc = GetDC(hwnd);
    hpalT = SelectPalette (hdc, hpalApp, FALSE);

    i = RealizePalette(hdc); /* i == entries that changed */

    SelectPalette (hdc, hpalT, FALSE);
    ReleaseDC(hwnd, hdc);

    /* If any palette entries changed, repaint the window. */

    if (i > 0)
        InvalidateRect(hwnd, NULL, TRUE);

    return i;
```

See Also

WM\_PALETTECHANGED, WM\_PALETTEISCHANGING

## WM\_QUERYOPEN (2.x)

### WM\_QUERYOPEN

The WM\_QUERYOPEN message is sent to a minimized window when the user requests that the window be restored to its preminimized size and position.

#### Parameters

This message has no parameters.

#### Returns

An application that processes this message should return a nonzero value if the icon can be opened or zero to prevent the icon from opened.

#### Comments

While processing this message, the application should not perform any action that would cause an activation or focus change (for example, creating a dialog box).

The DefWindowProc function returns nonzero when it processes this message.

#### See Also

DefWindowProc

## WM\_QUEUESYNC (3.1)

### WM\_QUEUESYNC

The WM\_QUEUESYNC message is sent by a computer-based training (CBT) application to separate user-input messages from other messages sent through the journal playback hook (WH\_JOURNALPLAYBACK).

#### Parameters

This message has no parameters.

#### Returns

A CBT application should return zero if it processes this message.

#### Comments

Whenever a CBT application uses the journal playback hook, the first and last messages rendered are WM\_QUEUESYNC. This allows the CBT application to intercept and examine user-initiated messages without doing so for events that it sends.

## WM\_QUIT (2.x)

```
WM_QUIT  
wExit = wParam; /* exit code */
```

The WM\_QUIT message indicates a request to terminate an application and is generated when the application calls the PostQuitMessage function. It causes the GetMessage function to return zero.

Parameter	Description
wExit	Value of wParam. Specifies the exit code given in the <u>PostQuitMessage</u> function.

### Returns

This message does not have a return value, because it causes the message loop to terminate before the message is sent to the application's window procedure.

### See Also

GetMessage, PostQuitMessage

## WM\_RBUTTONDOWNBLCLK (2.x)

```
WM_RBUTTONDOWNBLCLK
fwKeys = wParam; /* key flags */
xPos = LOWORD(lParam); /* horizontal position of cursor */
yPos = HIWORD(lParam); /* vertical position of cursor */
```

The WM\_RBUTTONDOWNBLCLK message is sent when the user double-clicks the right mouse button.

Parameter	Description												
fwKeys	Value of wParam. Indicates whether various virtual keys are down. This parameter can be any combination of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>MK_CONTROL</td><td>Set if CTRL key is down.</td></tr><tr><td>MK_LBUTTON</td><td>Set if left button is down.</td></tr><tr><td>MK_MBUTTON</td><td>Set if middle button is down.</td></tr><tr><td>MK_RBUTTON</td><td>Set if right button is down.</td></tr><tr><td>MK_SHIFT</td><td>Set if SHIFT key is down.</td></tr></tbody></table>	Value	Description	MK_CONTROL	Set if CTRL key is down.	MK_LBUTTON	Set if left button is down.	MK_MBUTTON	Set if middle button is down.	MK_RBUTTON	Set if right button is down.	MK_SHIFT	Set if SHIFT key is down.
Value	Description												
MK_CONTROL	Set if CTRL key is down.												
MK_LBUTTON	Set if left button is down.												
MK_MBUTTON	Set if middle button is down.												
MK_RBUTTON	Set if right button is down.												
MK_SHIFT	Set if SHIFT key is down.												
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.												
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.												

### Returns

An application should return zero if it processes this message.

### Comments

Only windows that have the CS\_DBLCLKS class style can receive WM\_RBUTTONDOWNBLCLK messages. Windows generates a WM\_RBUTTONDOWNBLCLK message when the user presses, releases, and again presses the right mouse button within the system's double-click time limit. Double-clicking the right mouse button actually generates four messages: a WM\_RBUTTONDOWN message, a WM\_RBUTTONUP message, the WM\_RBUTTONDOWNBLCLK message, and another WM\_RBUTTONUP message.

### See Also

WM\_RBUTTONDOWN, WM\_RBUTTONUP



## WM\_RBUTTONDOWN (2.x)

```
WM_RBUTTONDOWN
fwKeys = wParam; /* key flags */
xPos = LOWORD(lParam); /* horizontal position of cursor */
yPos = HIWORD(lParam); /* vertical position of cursor */
```

The WM\_RBUTTONDOWN message is sent when the user presses the right mouse button.

Parameter	Description										
fwKeys	Indicates whether various virtual keys are down. This parameter can be any combination of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>MK_CONTROL</td><td>Set if CTRL key is down.</td></tr><tr><td>MK_LBUTTON</td><td>Set if left mouse button is down.</td></tr><tr><td>MK_MBUTTON</td><td>Set if middle mouse button is down.</td></tr><tr><td>MK_SHIFT</td><td>Set if SHIFT key is down.</td></tr></tbody></table>	Value	Description	MK_CONTROL	Set if CTRL key is down.	MK_LBUTTON	Set if left mouse button is down.	MK_MBUTTON	Set if middle mouse button is down.	MK_SHIFT	Set if SHIFT key is down.
Value	Description										
MK_CONTROL	Set if CTRL key is down.										
MK_LBUTTON	Set if left mouse button is down.										
MK_MBUTTON	Set if middle mouse button is down.										
MK_SHIFT	Set if SHIFT key is down.										
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.										
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.										

### Returns

An application should return zero if it processes this message.

### See Also

WM\_RBUTTONDOWNBLCLK, WM\_RBUTTONUP

## WM\_RBUTTONDOWN (2.x)

WM\_RBUTTONDOWN

```
fwKeys = wParam; /* key flags */  
xPos = LOWORD(lParam); /* horizontal position of cursor */  
yPos = HIWORD(lParam); /* vertical position of cursor */
```

The WM\_RBUTTONDOWN message is sent when the user releases the right mouse button.

Parameter	Description										
fwKeys	Value of wParam. Indicates whether various virtual keys are down. This parameter can be any combination of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>MK_CONTROL</td><td>Set if CTRL key is down.</td></tr><tr><td>MK_LBUTTON</td><td>Set if left mouse button is down.</td></tr><tr><td>MK_MBUTTON</td><td>Set if middle mouse button is down.</td></tr><tr><td>MK_SHIFT</td><td>Set if SHIFT key is down.</td></tr></tbody></table>	Value	Description	MK_CONTROL	Set if CTRL key is down.	MK_LBUTTON	Set if left mouse button is down.	MK_MBUTTON	Set if middle mouse button is down.	MK_SHIFT	Set if SHIFT key is down.
Value	Description										
MK_CONTROL	Set if CTRL key is down.										
MK_LBUTTON	Set if left mouse button is down.										
MK_MBUTTON	Set if middle mouse button is down.										
MK_SHIFT	Set if SHIFT key is down.										
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.										
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor. The coordinate is relative to the upper-left corner of the window.										

### Returns

An application should return zero if it processes this message.

### See Also

[WM\\_RBUTTONDOWNBLCLK](#), [WM\\_RBUTTONDOWN](#)

## WM\_RENDERALLFORMATS (2.x)

### WM\_RENDERALLFORMATS

The WM\_RENDERALLFORMATS message is sent to the clipboard owner when the owner application is being destroyed.

#### Parameters

This message has no parameters.

#### Returns

An application should return zero if it processes this message.

#### Comments

The clipboard owner should render the data in all the formats it is capable of generating and pass a data handle for each format to the clipboard by calling the SetClipboardData function. This ensures that the clipboard contains valid data even though the application that rendered the data is destroyed. The application should call the OpenClipboard function before calling SetClipboardData and should call the CloseClipboard function afterward.

#### Example

In this example, the application sends a WM\_RENDERFORMAT message to itself for each clipboard format that the application supports:

```
case WM_RENDERALLFORMATS:
    OpenClipboard(hwnd);
    SendMessage(hwnd, WM_RENDERFORMAT, CF_DIB, 0L);
    SendMessage(hwnd, WM_RENDERFORMAT, CF_BITMAP, 0L);
    CloseClipboard();
    break;
```

#### See Also

CloseClipboard, OpenClipboard, SetClipboardData, WM\_RENDERFORMAT

## WM\_RENDERFORMAT (2.x)

```
WM_RENDERFORMAT
uFmt = (UINT) wParam; /* clipboard data format */
```

The WM\_RENDERFORMAT message is sent to the clipboard owner when a particular format with delayed rendering needs to be rendered. The receiver should render the data in that format and pass it to the clipboard by calling the SetClipboardData function.

Parameter	Description
uFmt	Specifies the data format. It can be any one of the formats described with the <u>SetClipboardData</u> function.

### Returns

An application should return zero if it processes this message.

### Comments

The application should not call the OpenClipboard and CloseClipboard functions while processing this message.

### Example

This example uses an application-defined function to render clipboard data. The function returns a data handle that is passed to the clipboard by the SetClipboardData function.

```
HANDLE hData;
```

```
case WM_RENDERFORMAT:
    if (hData = RenderFormat(wParam))
        SetClipboardData(wParam, hData);
    break;
```

### See Also

CloseClipboard, OpenClipboard, SetClipboardData, WM\_RENDERALLFORMATS

## WM\_SETCURSORS (2.x)

```
WM_SETCURSORS
hwndCursor = (HWND) wParam; /* handle of window with cursor */
nHittest = LOWORD(lParam); /* hit-test code */
wMouseMsg = HIWORD(lParam); /* mouse-message number */
```

The WM\_SETCURSORS message is sent to a window if mouse input is not captured and the mouse causes cursor movement within the window.

Parameter	Description
hwndCursor	Value of wParam. Specifies a handle to the window that contains the cursor.
nHittest	Value of the low-order word of lParam. Specifies the hit-test area code.
wMouseMsg	Value of the high-order word of lParam. Specifies the number of the mouse message.

### Returns

An application should return TRUE to halt further processing or zero to continue.

### Comments

If the nHittest parameter is HTERROR and the wMouseMsg parameter is a mouse button-down message, the [MessageBeep](#) function is called.

The [DefWindowProc](#) function passes the WM\_SETCURSORS message to a parent window before processing. If the parent window returns TRUE, further processing is halted. Passing the message to a window's parent window gives the parent window control over the cursor's setting in a child window. The DefWindowProc function also uses this message to set the cursor to a pointer if it is not in the client area or to the registered-class cursor if it is in the client area.

For a standard dialog box to set the cursor for one of its child window controls, it must force the [DefDlgProc](#) function to return TRUE in response to the WM\_SETCURSORS message. (DefDlgProc provides default processing for the standard dialog box class.) When DefDlgProc returns TRUE, the dialog procedure retains control over the cursor. When the dialog procedure processes the WM\_SETCURSORS message, it can return TRUE by using the [SetWindowLong](#) function and the [DWL\\_MSGRESULT](#) offset, as shown in the following example:

```
SetWindowLong(hwndDlg, DWL_MSGRESULT, MAKELONG(TRUE, 0));
```

### See Also

[DefWindowProc](#), [MessageBeep](#), [SetWindowLong](#)

## WM\_SETFOCUS (2.x)

WM\_SETFOCUS

```
hwnd = (HWND) wParam; /* handle of window losing focus */
```

The WM\_SETFOCUS message is sent after a window gains the input focus.

Parameter	Description
hwnd	Value of wParam. Contains the handle of the window that loses the input focus. (This parameter may be NULL.)

### Returns

An application should return zero if it processes this message.

### Comments

To display a caret, an application should call the appropriate caret functions at this point.

## WM\_SETFONT (3.0)

### WM\_SETFONT

```
wParam = (WPARAM) hfont; /* handle of the font */  
lParam = (LPARAM) MAKELONG((WORD) fRedraw, 0); /* redraw flag */
```

An application sends the WM\_SETFONT message to specify the font that a control is to use when drawing text.

Parameter	Description
hfont	Value of wParam. Specifies the handle of the font. If this parameter is NULL, the control will use the default system font to draw text.
fRedraw	Value of the low-order word of lParam. Specifies whether the control should be redrawn immediately upon setting the font. Setting the fRedraw parameter to TRUE causes the control to redraw itself.

### Returns

An application should return zero if it processes this message.

### Comments

The WM\_SETFONT message applies to all controls, not just those in dialog boxes.

The best time for the owner of a dialog box to set the font of the control is when it receives the WM\_INITDIALOG message. The application should call the [DeleteObject](#) function to delete the font when it is no longer needed--for example, after the control is destroyed.

The size of the control is not changed as a result of receiving this message. To prevent Windows from clipping text that does not fit within the boundaries of the control, the application should correct the size of the control window before changing the font.

Before Windows creates a dialog box with the DS\_SETFONT style, Windows sends the WM\_SETFONT message to the dialog box window before creating the controls. An application creates a dialog box with the DS\_SETFONT style by calling any of the following functions:

- [CreateDialogIndirect](#)
- [CreateDialogIndirectParam](#)
- [DialogBoxIndirect](#)
- [DialogBoxIndirectParam](#)

The DialogBoxHeader structure that the application passes to these functions must have the DS\_SETFONT style set and must contain the wPointSize and szFaceName members that define the font the dialog box will use to draw text.

For more information about the DialogBoxHeader structure, see the [Resource Format Overview](#).

### Example

This example changes the font used by controls in a dialog box to a font that is not bold.

```
HFONT hfontDlg;  
LOGFONT lFont;  
  
case WM_INITDIALOG:  
    /* Get dialog box font and create version that is not bold. */  
  
    hfontDlg = (HFONT) NULL;  
    if ((hfontDlg = (HFONT) SendMessage(hdlg, WM_GETFONT, 0, 0L)) {  
        if (GetObject(hfontDlg, sizeof(LOGFONT), (LPSTR) &lFont)) {  
            lFont.lfWeight = FW_NORMAL;  
            if (hfontDlg = CreateFontIndirect((LPLOGFONT) &lFont)) {  
                SendDlgItemMessage(hdlg, ID_CTRL1, WM_SETFONT,  
(WPARAM) hfontDlg, 0);  
                SendDlgItemMessage(hdlg, ID_CTRL2, WM_SETFONT,  
(WPARAM) hfontDlg, 0);  
                .  
                . /* Set font for remaining controls. */  
                .  
            }  
        }  
    }  
}
```

```
}  
}  
return TRUE;
```

See Also

CreateDialogIndirect, CreateDialogIndirectParam, DeleteObject, DialogBoxIndirect,  
DialogBoxIndirectParam, WM\_INITDIALOG, WM\_SETFONT



## WM\_SETREDRAW (2.x)

WM\_SETREDRAW

```
wParam = (WPARAM) fRedraw; /* state of redraw flag */  
lParam = 0L; /* not used, must be zero */
```

An application sends a WM\_SETREDRAW message to a window to allow changes in that window to be redrawn or to prevent changes in that window from being redrawn.

Parameter	Description
fRedraw	Value of wParam. Specifies the state of the redraw flag. If this parameter is nonzero, the redraw flag is set. If this parameter is zero, the flag is cleared.

### Returns

An application should return zero if it processes this message.

### Comments

This message sets or clears the redraw flag. If the redraw flag is cleared, the contents of the specified window will not be updated after each change, and the window will not be repainted until the redraw flag is set. For example, an application that needs to add several items to a list box can clear the redraw flag, add the items, and then set the redraw flag. Finally, the application can call the InvalidateRect function to cause the list box to be repainted.

## WM\_SETTEXT (2.x)

### WM\_SETTEXT

```
wParam = 0; /* not used, must be zero */
lParam = (LPARAM) (LPCSTR) pszText; /* address of window-text string */
/
```

An application sends a WM\_SETTEXT message to set the text of a window.

Parameter	Description
pszText	Value of lParam. Points to a null-terminated string that is the window text.

### Returns

The return value is LB\_ERRSPACE (for a list box) or CB\_ERRSPACE (for a combo box) if insufficient space is available to set the text in the edit control. It is CB\_ERR if this message is sent to a combo box without an edit control.

### Comments

For an edit control, the text to be set is the contents of the edit control. For a combo box, the text is the contents of the edit-control (or static-text) portion of the combo box. For a button, the text is the button name. For other windows, the text is the window title.

This message does not change the current selection in the list box of a combo box. An application should use the CB\_SELECTSTRING message to select the item in the list box that matches the text in the edit control.

### See Also

WM\_GETTEXT

## WM\_SHOWWINDOW (2.x)

```
WM_SHOWWINDOW
fShow = (BOOL) wParam;    /* show/hide flag */
fnStatus = LOWORD(lParam); /* status flag    */
```

The WM\_SHOWWINDOW message is sent to a window when it is about to be hidden or shown. A window is hidden or shown when the ShowWindow function is called; when an overlapped window is maximized or restored; or when an overlapped or pop-up window is minimized or displayed on the screen. When an overlapped window is minimized, all pop-up windows associated with that window are hidden.

Parameter	Description						
fShow	Value of wParam. Specifies whether a window is being shown. It is TRUE if the window is being shown; it is FALSE if the window is being hidden.						
fnStatus	Value of the low-order word of lParam. Specifies the status of the window being shown. The fnStatus parameter is zero if the message is sent because of a <u>ShowWindow</u> function call; otherwise, fnStatus is one of the following values:						
	<table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>SW_PARENTCLOSING</td><td>Parent window is being minimized, or a pop-up window is being hidden.</td></tr><tr><td>SW_PARENTOPENING</td><td>Parent window is opening (being displayed) or a pop-up window is being shown.</td></tr></tbody></table>	Value	Description	SW_PARENTCLOSING	Parent window is being minimized, or a pop-up window is being hidden.	SW_PARENTOPENING	Parent window is opening (being displayed) or a pop-up window is being shown.
Value	Description						
SW_PARENTCLOSING	Parent window is being minimized, or a pop-up window is being hidden.						
SW_PARENTOPENING	Parent window is opening (being displayed) or a pop-up window is being shown.						

### Returns

An application should return zero if it processes this message.

### Comments

The DefWindowProc function hides or shows the window as specified by the message.

The WM\_SHOWWINDOW message is not sent under the following circumstances:

- When a main window is created with the WS\_MAXIMIZE or WS\_MINIMIZE style
- When the SW\_SHOWNORMAL flag is specified in the call to the ShowWindow function

### See Also

DefWindowProc, ShowWindow

## WM\_SIZE (2.x)

### WM\_SIZE

```
fwSizeType = wParam; /* sizing-type flag */  
nWidth = LOWORD(lParam); /* width of client area */  
nHeight = HIWORD(lParam); /* height of client area */
```

The WM\_SIZE message is sent to a window after its size has changed.

Parameter	Description												
fwSizeType	Value of wParam. Specifies the type of resizing requested. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>SIZE_MAXIMIZED</td><td>Window has been maximized.</td></tr><tr><td>SIZE_MINIMIZED</td><td>Window has been minimized.</td></tr><tr><td>SIZE_RESTORED</td><td>Window has been resized, but neither SIZE_MINIMIZED nor SIZE_MAXIMIZED applies.</td></tr><tr><td>SIZE_MAXHIDE</td><td>Message is sent to all pop-up windows when some other window is maximized.</td></tr><tr><td>SIZE_MAXSHOW</td><td>Message is sent to all pop-up windows when some other window has been restored to its former size.</td></tr></tbody></table>	Value	Description	SIZE_MAXIMIZED	Window has been maximized.	SIZE_MINIMIZED	Window has been minimized.	SIZE_RESTORED	Window has been resized, but neither SIZE_MINIMIZED nor SIZE_MAXIMIZED applies.	SIZE_MAXHIDE	Message is sent to all pop-up windows when some other window is maximized.	SIZE_MAXSHOW	Message is sent to all pop-up windows when some other window has been restored to its former size.
Value	Description												
SIZE_MAXIMIZED	Window has been maximized.												
SIZE_MINIMIZED	Window has been minimized.												
SIZE_RESTORED	Window has been resized, but neither SIZE_MINIMIZED nor SIZE_MAXIMIZED applies.												
SIZE_MAXHIDE	Message is sent to all pop-up windows when some other window is maximized.												
SIZE_MAXSHOW	Message is sent to all pop-up windows when some other window has been restored to its former size.												
nWidth	Value of the low-order word of lParam. Specifies the new width of the client area.												
nHeight	Value of the high-order word of lParam. Specifies the new height of the client area.												

### Returns

An application should return zero if it processes this message.

### Comments

If the SetScrollPos or MoveWindow function is called for a child window as a result of the WM\_SIZE message, the fRepaint parameter should be nonzero to cause the window to be repainted.

### See Also

MoveWindow, SetScrollPos

## WM\_SIZECLIPBOARD (2.x)

WM\_SIZECLIPBOARD

```
hwndViewer = (HWND) wParam; /* handle of clipboard viewer */  
hglb = (HGLOBAL) LOWORD(lParam); /* handle of global object */
```

The WM\_SIZECLIPBOARD message is sent by the clipboard viewer to the clipboard owner when the clipboard contains data with the CF\_OWNERDISPLAY attribute and the size of the client area of the clipboard-viewer window has changed.

Parameter	Description
hwndViewer	Value of wParam. Identifies the clipboard-application window.
hglb	Value of the low-order word of lParam. Identifies a global memory object that contains a <u>RECT</u> data structure. The structure specifies the area that the clipboard owner should paint.

### Returns

An application should return zero if it processes this message.

### Comments

A WM\_SIZECLIPBOARD message is sent with a null rectangle (0,0,0,0) as the new size when the clipboard application is about to be destroyed or minimized. This permits the clipboard owner to free its display resources.

An application must use the GlobalLock function to lock the memory that contains the RECT data structure. The application should unlock that memory by using the GlobalUnlock function before it yields or returns control.

### See Also

GlobalLock, GlobalUnlock, SetClipboardData, SetClipboardViewer, RECT

## WM\_SPOOLERSTATUS (3.0)

WM\_SPOOLERSTATUS

```
fwJobStatus = wParam; /* job-status flag*/
```

```
cJobsLeft = LOWORD(lParam); /* number of jobs remaining */
```

The WM\_SPOOLERSTATUS message is sent from Print Manager whenever a job is added to or removed from the Print Manager queue.

Parameter	Description
fwJobStatus	Value of wParam. Specifies the SP_JOBSTATUS flag.
cJobsLeft	Value of the low-order word of lParam. Specifies the number of jobs remaining in the Print Manager queue.

### Returns

An application should return zero if it processes this message.

### Comments

This message is for informational purposes only.

## WM\_SYSCHAR (2.x)

```
WM_SYSCHAR
wKeyCode = wParam; /* ASCII key code */
dwKeyData = lParam; /* key data */
```

The WM\_SYSCHAR message is sent to the window with the input focus when a WM\_SYSKEYUP and a WM\_SYSKEYDOWN message are translated. It specifies the virtual-key code of the System-menu key. (The System menu is sometimes referred to as the Control menu.)

Parameter	Description																		
wKeyCode	Value of wParam. Specifies the ASCII-character key code of a System-menu key.																		
dwKeyData	Value of lParam. Specifies the repeat count, scan code, extended key, context code, previous key state, and key-transition state, as shown in the following table:																		
	<table><thead><tr><th>Bit</th><th>Description</th></tr></thead><tbody><tr><td>0-15</td><td>Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.</td></tr><tr><td>16-23</td><td>Specifies the scan code. The value depends on the original equipment manufacturer (OEM).</td></tr><tr><td>24</td><td>Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.</td></tr><tr><td>25-26</td><td>Not used.</td></tr><tr><td>27-28</td><td>Used internally by Windows.</td></tr><tr><td>29</td><td>Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.</td></tr><tr><td>30</td><td>Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.</td></tr><tr><td>31</td><td>Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.</td></tr></tbody></table>	Bit	Description	0-15	Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.	16-23	Specifies the scan code. The value depends on the original equipment manufacturer (OEM).	24	Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.	25-26	Not used.	27-28	Used internally by Windows.	29	Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.	30	Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.	31	Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.
Bit	Description																		
0-15	Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.																		
16-23	Specifies the scan code. The value depends on the original equipment manufacturer (OEM).																		
24	Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.																		
25-26	Not used.																		
27-28	Used internally by Windows.																		
29	Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.																		
30	Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.																		
31	Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.																		

### Returns

An application should return zero if it processes this message.

### Comments

When the context code is zero, the message can be passed to the TranslateAccelerator function, which will handle it as though it were a normal key message instead of a System-menu key message. This allows accelerator keys to be used with the active window even if the active window does not have the input focus.

For IBM Enhanced 101- and 102-key keyboards, enhanced keys are the right ALT key and the right CTRL key on the main section of the keyboard; the INS, DEL, HOME, END, PAGE UP, PAGE DOWN, and arrow keys in the clusters to the left of the numeric keypad; and the division (/) and ENTER keys on the numeric keypad. Some other keyboards may support the extended-key bit in the lParam parameter.

### See Also

TranslateAccelerator, WM\_SYSKEYDOWN, WM\_SYSKEYUP

## WM\_SYSCOLORCHANGE (2.x)

### WM\_SYSCOLORCHANGE

The WM\_SYSCOLORCHANGE message is sent to all top-level windows when a change is made in the system color setting.

#### Parameters

This message has no parameters.

#### Returns

An application should return zero if it processes this message.

#### Comments

Windows sends a WM\_PAINT message to any window that is affected by a system color change.

Applications that have brushes that use the existing system colors should delete those brushes and re-create them by using the new system colors.

#### See Also

SetSysColors, WM\_PAINT



## WM\_SYSCOMMAND (2.x)

```
WM_SYSCOMMAND
wCmdType = wParam; /* command value */
xPos = LOWORD(lParam); /* horizontal position of cursor */
yPos = HIWORD(lParam); /* vertical position of cursor */
```

The WM\_SYSCOMMAND message is sent when the user selects a command from the System menu (sometimes referred to as the Control menu) or when the user selects the Maximize button or the Minimize button.

Parameter	Description																																
wCmdType	Value of wParam. Specifies the type of system command requested. This parameter can be one of the following values:																																
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>SC_CLOSE</td><td>Close the window.</td></tr><tr><td>SC_HOTKEY</td><td>Activate the window associated with the application-specified hot key. The low-order word of lParam identifies the window to activate.</td></tr><tr><td>SC_HSCROLL</td><td>Scroll horizontally.</td></tr><tr><td>SC_KEYMENU</td><td>Retrieve a menu through a keystroke.</td></tr><tr><td>SC_MAXIMIZE (or SC_ZOOM)</td><td>Maximize the window.</td></tr><tr><td>SC_MINIMIZE (or SC_ICON)</td><td>Minimize the window.</td></tr><tr><td>SC_MOUSEMENU</td><td>Retrieve a menu through a mouse click.</td></tr><tr><td>SC_MOVE</td><td>Move the window.</td></tr><tr><td>SC_NEXTWINDOW</td><td>Move to the next window.</td></tr><tr><td>SC_PREVWINDOW</td><td>Move to the previous window.</td></tr><tr><td>SC_RESTORE</td><td>Restore window to normal position and size.</td></tr><tr><td>SC_SCREENSAVE</td><td>Execute the screen-saver application specified in the [boot] section of the SYSTEM.INI file.</td></tr><tr><td>SC_SIZE</td><td>Size the window.</td></tr><tr><td>SC_TASKLIST</td><td>Execute or activate the Windows Task Manager application.</td></tr><tr><td>SC_VSCROLL</td><td>Scroll vertically.</td></tr></tbody></table>	Value	Meaning	SC_CLOSE	Close the window.	SC_HOTKEY	Activate the window associated with the application-specified hot key. The low-order word of lParam identifies the window to activate.	SC_HSCROLL	Scroll horizontally.	SC_KEYMENU	Retrieve a menu through a keystroke.	SC_MAXIMIZE (or SC_ZOOM)	Maximize the window.	SC_MINIMIZE (or SC_ICON)	Minimize the window.	SC_MOUSEMENU	Retrieve a menu through a mouse click.	SC_MOVE	Move the window.	SC_NEXTWINDOW	Move to the next window.	SC_PREVWINDOW	Move to the previous window.	SC_RESTORE	Restore window to normal position and size.	SC_SCREENSAVE	Execute the screen-saver application specified in the [boot] section of the SYSTEM.INI file.	SC_SIZE	Size the window.	SC_TASKLIST	Execute or activate the Windows Task Manager application.	SC_VSCROLL	Scroll vertically.
Value	Meaning																																
SC_CLOSE	Close the window.																																
SC_HOTKEY	Activate the window associated with the application-specified hot key. The low-order word of lParam identifies the window to activate.																																
SC_HSCROLL	Scroll horizontally.																																
SC_KEYMENU	Retrieve a menu through a keystroke.																																
SC_MAXIMIZE (or SC_ZOOM)	Maximize the window.																																
SC_MINIMIZE (or SC_ICON)	Minimize the window.																																
SC_MOUSEMENU	Retrieve a menu through a mouse click.																																
SC_MOVE	Move the window.																																
SC_NEXTWINDOW	Move to the next window.																																
SC_PREVWINDOW	Move to the previous window.																																
SC_RESTORE	Restore window to normal position and size.																																
SC_SCREENSAVE	Execute the screen-saver application specified in the [boot] section of the SYSTEM.INI file.																																
SC_SIZE	Size the window.																																
SC_TASKLIST	Execute or activate the Windows Task Manager application.																																
SC_VSCROLL	Scroll vertically.																																
xPos	Value of the low-order word of lParam. Specifies the x-coordinate of the cursor, if a System-menu command is chosen with the mouse. Otherwise, this parameter is not used.																																
yPos	Value of the high-order word of lParam. Specifies the y-coordinate of the cursor, if a System-menu command is chosen with the mouse. Otherwise, this parameter is not used.																																

### Returns

An application should return zero if it processes this message.

### Comments

The DefWindowProc function carries out the System-menu request for the predefined actions specified in the preceding table.

In WM\_SYSCOMMAND messages, the four low-order bits of the wCmdType parameter are used internally by Windows. When an application tests the value of wCmdType, it must combine the value 0xFFFF0 with the wCmdType value by using the bitwise AND operator to obtain the correct result.

The menu items in a System menu can be modified by using the GetSystemMenu, AppendMenu, InsertMenu, and ModifyMenu functions. Applications that modify the System menu must process WM\_SYSCOMMAND messages. Any WM\_SYSCOMMAND messages not handled by the application must be passed to the DefWindowProc function. Any command values added by an application must be processed by the application and cannot be passed to DefWindowProc.

An application can carry out any system command at any time by passing a WM\_SYSCOMMAND message to the DefWindowProc function.

Accelerator keystrokes that are defined to select items from the System menu are translated into WM\_SYSCOMMAND messages; all other accelerator key strokes are translated into WM\_COMMAND messages.

See Also

AppendMenu, DefWindowProc, GetSystemMenu, InsertMenu, ModifyMenu, WM\_COMMAND

### Windows 3.1 changes

The following system-command values have been added:

Value	Meaning
SC_HOTKEY	Activate the window associated with the application-specified hot key. The low-order word of lParam identifies the window to activate.
SC_SCREENSAVE	Executes the screen-save application specified in the Desktop section of Control Panel.

## WM\_SYSDEADCHAR (2.x)

WM\_SYSDEADCHAR

```
wDeadKey = wParam; /* dead-key character */  
cRepeat = (int) LOWORD(lParam); /* repeat count */  
cAutoRepeat = HIWORD(lParam); /* auto-repeat count */
```

The WM\_SYSDEADCHAR message is sent to the window with the input focus when WM\_SYSKEYUP and WM\_SYSKEYDOWN messages are translated. It specifies the character value of a dead key.

Parameter	Description
wDeadKey	Value of wParam. Specifies the dead-key character value.
cRepeat	Value of the low-order word of lParam. Specifies the repeat count.
cAutoRepeat	Value of the high-order word of lParam. Specifies the auto-repeat count.

Returns

An application should return zero if it processes this message.

See Also

WM\_SYSKEYDOWN, WM\_SYSKEYUP

## WM\_SYSKEYDOWN (2.x)

```
WM_SYSKEYDOWN
wVkey = wParam; /* virtual-key code */
dwKeyData = lParam; /* key data */
```

The WM\_SYSKEYDOWN message is sent to the window with the input focus when the user holds down the ALT key and then presses another key. If no window currently has the input focus, the WM\_SYSKEYDOWN message is sent to the active window. The window that receives the message can distinguish between these two contexts by checking the context code in the dwKeyData parameter.

Parameter	Description
wVkey	Value of wParam. Specifies the virtual-key code of the key being pressed.
dwKeyData	Value of lParam. Specifies the repeat count, scan code, extended key, context code, previous key state, and key-transition state, as shown in the following table:
Bit	Description
0-15	Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.
16-23	Specifies the scan code. The value depends on the original equipment manufacturer (OEM).
24	Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.
25-26	Not used.
27-28	Used internally by Windows.
29	Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.
30	Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.
31	Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.

For WM\_SYSKEYDOWN messages, the value of bit 29 (context code) is 1 if the ALT key is down while the key is pressed; it is 0 if the message is sent to the active window because no window has the input focus. The value of bit 31 (key-transition state) is 0.

### Returns

An application should return zero if it processes this message.

### Comments

When the context code is zero, the message can be passed to the [TranslateAccelerator](#) function, which will handle it as though it were a normal key message instead of a system-key message. This allows accelerator keys to be used with the active window even if the active window does not have the input focus.

Because of the autorepeat feature, more than one WM\_SYSKEYDOWN message may occur before a WM\_SYSKEYUP message is sent. The previous key state (bit 30) can be used to determine whether the WM\_SYSKEYDOWN message indicates the first down transition or a repeated down transition.

For IBM Enhanced 101- and 102-key keyboards, enhanced keys are the right ALT key and the right CTRL key on the main section of the keyboard; the INS, DEL, HOME, END, PAGE UP, PAGE DOWN, and arrow keys in the clusters to the left of the numeric keypad; and the division (/) and ENTER keys on the numeric keypad. Some other keyboards may support the extended-key bit in the lParam parameter.

### See Also

[TranslateAccelerator](#), [WM\\_SYSKEYUP](#)

## WM\_SYSKEYUP (2.x)

```
WM_SYSKEYUP
wVkey = wParam; /* virtual-key code */
dwKeyData = lParam; /* key data */
```

The WM\_SYSKEYUP message is sent to the window with the input focus when the user releases a key that was pressed while the ALT key was held down. If no window currently has the input focus, the WM\_SYSKEYUP message is sent to the active window. The window that receives the message can distinguish between these two contexts by checking the context code in the lParam parameter.

Parameter	Description
wVkey	Value of wParam. Specifies the virtual-key code of the key being pressed.
dwKeyData	Value of lParam. Specifies the repeat count, scan code, extended key, context code, previous key state, and key-transition state, as shown in the following table:
Bit	Description
0-15	Specifies the repeat count. The value is the number of times the keystroke is repeated as a result of the user holding down the key.
16-23	Specifies the scan code. The value depends on the original equipment manufacturer (OEM).
24	Specifies whether the key is an extended key, such as a function key or a key on the numeric keypad. The value is 1 if it is an extended key; otherwise, it is 0.
25-26	Not used.
27-28	Used internally by Windows.
29	Specifies the context code. The value is 1 if the ALT key is held down while the key is pressed; otherwise, the value is 0.
30	Specifies the previous key state. The value is 1 if the key is down before the message is sent, or it is 0 if the key is up.
31	Specifies the key-transition state. The value is 1 if the key is being released, or it is 0 if the key is being pressed.

For WM\_SYSKEYUP messages, the value of bit 29 (context code) is 1 if the ALT key is down while the key is pressed; it is 0 if the message is sent to the active window because no window has the input focus. The value of bit 31 (key-transition state) is 1.

### Returns

An application should return zero if it processes this message.

### Comments

When the context code is zero, the message can be passed to the [TranslateAccelerator](#) function, which will handle it as though it were a normal key message instead of a system-key message. This allows accelerator keys to be used with the active window even if the active window does not have the input focus.

For IBM Enhanced 101- and 102-key keyboards, enhanced keys are the right ALT key and the right CTRL key on the main section of the keyboard; the INS, DEL, HOME, END, PAGE UP, PAGE DOWN, and arrow keys in the clusters to the left of the numeric keypad; and the division (/) and ENTER keys on the numeric keypad. Some other keyboards may support the extended-key bit in the lParam parameter.

For non-U.S. Enhanced 102-key keyboards, the right ALT key is handled as the CTRL+ALT key combination. The following list shows the messages that result when the user presses and releases this key, in the sequence they occur:

- 1 WM\_KEYDOWN VK\_CONTROL
- 2 WM\_KEYDOWN VK\_MENU
- 3 WM\_KEYUP VK\_CONTROL
- 4 WM\_SYSKEYUP VK\_MENU

### See Also

[TranslateAccelerator](#), [WM\\_SYSKEYDOWN](#)

## WM\_SYSTEMERROR (3.1)

WM\_SYSTEMERROR

```
wErrSpec = wParam; /* specifies when error occurred */
```

The WM\_SYSTEMERROR message is sent when the Windows kernel encounters an error but cannot display the system-error message box.

Parameter	Description
wErrSpec	Value of wParam. Specifies when the error occurred. Currently, the only valid value is 1, indicating that the error occurred when a task or library was terminating.

### Returns

An application should return zero if it processes this message.

### Comments

A shell application should process this message, displaying a message box that indicates an error has occurred.

## WM\_TIMECHANGE (2.x)

WM\_TIMECHANGE

```
wParam = 0; /* not use, must be zero */  
lParam = 0L; /* not use, must be zero */
```

An application sends the WM\_TIMECHANGE message to all top-level windows after changing the system time.

Parameters

This message has no parameters.

Returns

An application should return zero if it processes this message.

Comments

Any application that changes the system time should send this message to all top-level windows. To send the WM\_TIMECHANGE message to all top-level windows, an application can use the SendMessage function with the hwnd parameter set to HWND\_BROADCAST.

See Also

SendMessage



## WM\_TIMER (2.x)

```
WM_TIMER
wTimerID = wParam; /* timer identifier */
tmprc = (TIMERPROC FAR*) lParam; /* address of timer callback*/
```

The WM\_TIMER message is posted to the installing application's message queue or sent to the appropriate TimerProc callback function after each interval specified in the SetTimer function used to install a timer.

Parameter	Description
wTimerID	Value of wParam. Specifies the identifier of the timer.
tmprc	Value of lParam. Points to a callback function that was passed to the <u>SetTimer</u> function when the timer was installed. If the tmprc parameter is not NULL, the system passes the WM_TIMER message to the specified callback function rather than posting the message to the application's message queue.

### Returns

An application should return zero if it processes this message.

### Comments

The DispatchMessage function sends this message when no other messages are in the application's message queue.

### Example

This example uses the WM\_TIMER message to create a blinking effect for a line of text:

```
DWORD dwXYVal;
WORD wXVal, wYVal;
char szMessage[16];

case WM_TIMER:
    hdc = GetDC(hwnd);
    dwXYVal = GetTextExtent(hdc, (LPCSTR) szMessage,
        lstrlen(szMessage));
    wXVal = LOWORD(dwXYVal);
    wYVal = HIWORD(dwXYVal);
    PatBlt(hdc, 10, 10, (int) wXVal, (int) wYVal, PATINVERT);
    ReleaseDC(hwnd, hdc);
    ValidateRect(hwnd, NULL);
    break;
```

### See Also

SetTimer, TimerProc

## WM\_UNDO (2.x)

### WM\_UNDO

An application sends the WM\_UNDO message to an edit control to undo the last operation. When this message is sent to an edit control, the previously deleted text is restored or the previously added text is deleted.

### Parameters

This message has no parameters.

### Returns

The return value is nonzero if the operation is successful, or it is zero if an error occurs.

### See Also

WM\_CLEAR, WM\_COPY, WM\_CUT, WM\_PASTE

## WM\_USER (2.x)

### WM\_USER

WM\_USER is a constant used by applications to help define private messages.

#### Comments

The WM\_USER constant is used to distinguish between message values that are reserved for use by Windows and values that can be used by an application to send messages within a private window class. There are four ranges of message numbers:

Range	Meaning
0 through WM_USER - 1	Messages reserved for use by Windows.
WM_USER through 0x7FFF	Integer messages for use by private window classes.
0x8000 through 0xBFFF	Messages reserved for use by Windows.
0xC000 through 0xFFFF	String messages for use by applications.

Message numbers in the first range (0 through WM\_USER - 1) are defined by Windows. Values in this range that are not explicitly defined are reserved for future use by Windows. This topic describes messages in this range.

Message numbers in the second range (WM\_USER through 0x7FFF) can be defined and used by an application to send messages within a private window class. These values cannot be used to define messages that are meaningful throughout an application, because some predefined window classes already define values in this range. For example, such predefined control classes as BUTTON, EDIT, LISTBOX, and COMBOBOX may use these values. Messages in this range should not be sent to other applications unless the applications have been designed to exchange messages and to attach the same meaning to the message numbers.

Message numbers in the third range (0x8000 through 0xBFFF) are reserved for future use by Windows.

Message numbers in the fourth range (0xC000 through 0xFFFF) are defined at run time when an application calls the RegisterWindowMessage function to obtain a message number for a string. All applications that register the same string can use the associated message number for exchanging messages. The actual message number, however, is not a constant and cannot be assumed to be the same in different Windows sessions.

See Also

[RegisterWindowMessage](#)

## WM\_VKEYTOITEM (3.0)

```
WM_VKEYTOITEM
wVkey = wParam; /* virtual-key code */
hwndLB = (HWND) LOWORD(lParam); /* handle of the list box */
nCaretPos = HIWORD(lParam); /* caret position */
```

The WM\_VKEYTOITEM message is sent by a list box with the LBS\_WANTKEYBOARDINPUT style to its owner in response to a WM\_KEYDOWN message.

Parameter	Description
wVkey	Value of wParam. Specifies the virtual-key code of the key that the user pressed.
hwndLB	Value of the low-order word of lParam. Identifies the list box.
nCaretPos	Value of the high-order word of lParam. Specifies the current position of the caret.

### Returns

The return value specifies the action that the application performed in response to the message. A return value of -2 indicates that the application handled all aspects of selecting the item and requires no further action by the list box. A return value of -1 indicates that the list box should perform the default action in response to the keystroke. A return value of 0 or greater specifies the zero-based index of an item in the list box and indicates that the list box should perform the default action for the keystroke on the given item.

### Comments

Only list boxes that have the LBS\_HASSTRINGS style can receive this message.

### See Also

WM\_CHARTOITEM, WM\_KEYDOWN

## WM\_VSCROLL (2.x)

```
WM_VSCROLL
wScrollCode = wParam; /* scroll bar code */
nPos = LOWORD(lParam); /* current scroll box position */
hwndCtl = (HWND) HIWORD(lParam); /* handle of the control */
```

The WM\_VSCROLL message is sent to a window when the user clicks the window's vertical scroll bar.

Parameter	Description																				
wScrollCode	Value of wParam. Specifies a scroll bar code that indicates the user's scrolling request. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>SB_BOTTOM</td><td>Scroll to bottom.</td></tr><tr><td>SB_ENDSCROLL</td><td>End scroll.</td></tr><tr><td>SB_LINEDOWN</td><td>Scroll one line down.</td></tr><tr><td>SB_LINEUP</td><td>Scroll one line up.</td></tr><tr><td>SB_PAGEDOWN</td><td>Scroll one page down.</td></tr><tr><td>SB_PAGEUP</td><td>Scroll one page up.</td></tr><tr><td>SB_THUMBPOSITION</td><td>Scroll to absolute position. The current position is specified by the nPos parameter.</td></tr><tr><td>SB_THUMBTRACK</td><td>Drag scroll box (thumb) to specified position. The current position is specified by the nPos parameter.</td></tr><tr><td>SB_TOP</td><td>Scroll to top.</td></tr></tbody></table>	Value	Description	SB_BOTTOM	Scroll to bottom.	SB_ENDSCROLL	End scroll.	SB_LINEDOWN	Scroll one line down.	SB_LINEUP	Scroll one line up.	SB_PAGEDOWN	Scroll one page down.	SB_PAGEUP	Scroll one page up.	SB_THUMBPOSITION	Scroll to absolute position. The current position is specified by the nPos parameter.	SB_THUMBTRACK	Drag scroll box (thumb) to specified position. The current position is specified by the nPos parameter.	SB_TOP	Scroll to top.
Value	Description																				
SB_BOTTOM	Scroll to bottom.																				
SB_ENDSCROLL	End scroll.																				
SB_LINEDOWN	Scroll one line down.																				
SB_LINEUP	Scroll one line up.																				
SB_PAGEDOWN	Scroll one page down.																				
SB_PAGEUP	Scroll one page up.																				
SB_THUMBPOSITION	Scroll to absolute position. The current position is specified by the nPos parameter.																				
SB_THUMBTRACK	Drag scroll box (thumb) to specified position. The current position is specified by the nPos parameter.																				
SB_TOP	Scroll to top.																				
nPos	Value of the low-order word of lParam. Specifies the current position of the scroll box if wScrollCode is SB_THUMBPOSITION or SB_THUMBTRACK; otherwise, this parameter is not used.																				
hwndCtl	Value of the high-order word of lParam. Identifies the control if WM_VSCROLL is sent by a scroll bar. If WM_VSCROLL is sent as a result of the user clicking a pop-up window's scroll bar, the high-order word is not used.																				

### Returns

An application should return zero if it processes this message.

### Comments

The SB\_THUMBTRACK message typically is used by applications that give some feedback while the scroll box is being dragged.

If an application scrolls the contents of the window, it must also reset the position of the scroll box by using the [SetScrollPos](#) function.

### See Also

[SetScrollPos](#), [WM\\_HSCROLL](#)

## WM\_VSCROLLCLIPBOARD (2.x)

### WM\_VSCROLLCLIPBOARD

```
hwndViewer = (HWND) wParam; /* handle of clipboard viewer */
wScrollCode = LOWORD(lParam); /* scroll bar code */
wThumbPos = HIWORD(lParam); /* scroll box position */
```

The WM\_HSCROLLCLIPBOARD message is sent by the clipboard viewer to the clipboard owner when the clipboard data has the CF\_OWNERDISPLAY format and there is an event in the clipboard viewer's vertical scroll bar. The owner should scroll the clipboard image, invalidate the appropriate section, and update the scroll bar values.

Parameter	Description																		
hwndViewer	Value of wParam. Specifies a handle to a clipboard-viewer window.																		
wScrollCode	Value of the low-order word of lParam. Specifies one of the following scroll bar values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>SB_BOTTOM</td><td>Scroll to lower right.</td></tr><tr><td>SB_ENDSCROLL</td><td>End scroll.</td></tr><tr><td>SB_LINEDOWN</td><td>Scroll one line down.</td></tr><tr><td>SB_LINEUP</td><td>Scroll one line up.</td></tr><tr><td>SB_PAGEDOWN</td><td>Scroll one page down.</td></tr><tr><td>SB_PAGEUP</td><td>Scroll one page up.</td></tr><tr><td>SB_THUMBPOSITION</td><td>Scroll to absolute position.</td></tr><tr><td>SB_TOP</td><td>Scroll to upper left.</td></tr></tbody></table>	Value	Description	SB_BOTTOM	Scroll to lower right.	SB_ENDSCROLL	End scroll.	SB_LINEDOWN	Scroll one line down.	SB_LINEUP	Scroll one line up.	SB_PAGEDOWN	Scroll one page down.	SB_PAGEUP	Scroll one page up.	SB_THUMBPOSITION	Scroll to absolute position.	SB_TOP	Scroll to upper left.
Value	Description																		
SB_BOTTOM	Scroll to lower right.																		
SB_ENDSCROLL	End scroll.																		
SB_LINEDOWN	Scroll one line down.																		
SB_LINEUP	Scroll one line up.																		
SB_PAGEDOWN	Scroll one page down.																		
SB_PAGEUP	Scroll one page up.																		
SB_THUMBPOSITION	Scroll to absolute position.																		
SB_TOP	Scroll to upper left.																		
wThumbPos	Value of the high-order word of lParam. Specifies the scroll box position if the scroll bar code is SB_THUMBPOSITION; otherwise, the high-order word is not used.																		

### Returns

An application should return zero if it processes this message.

### Comments

The clipboard owner should use the InvalidateRect function or repaint the window as needed. The scroll bar position should also be reset.

### See Also

InvalidateRect, WM\_HSCROLLCLIPBOARD

## WM\_WINDOWPOSCHANGED (3.1)

WM\_WINDOWPOSCHANGED

```
pwp = (const WINDOWPOS FAR*) lParam; /* structure address*/
```

The WM\_WINDOWPOSCHANGED message is sent to a window whose size, position, or z-order has changed as a result of a call to SetWindowPos or another window-management function.

Parameter	Description
pwp	Value of lParam. Points to a <u>WINDOWPOS</u> data structure that contains information about the window's new size and position.

### Returns

An application should return zero if it processes this message.

### Comments

The DefWindowProc function, when it processes the WM\_WINDOWPOSCHANGED message, sends the WM\_SIZE and WM\_MOVE messages to the window. These messages are not sent if an application handles the WM\_WINDOWPOSCHANGED message without calling DefWindowProc. It is more efficient to perform any move or size change processing during the WM\_WINDOWPOSCHANGED message without calling DefWindowProc.

### See Also

WM\_MOVE, WM\_SIZE, WM\_WINDOWPOSCHANGING, EndDeferWindowPos, SetWindowPos

## WM\_WINDOWPOSCHANGING (3.1)

WM\_WINDOWPOSCHANGING

```
pwp = (WINDOWPOS FAR*) lParam; /* address of WINDOWPOS structure */
```

The WM\_WINDOWPOSCHANGING message is sent to a window whose size, position, or z-order is about to change as a result of a call to SetWindowPos or another window-management function.

Parameter	Description
pwp	Value of lParam. Points to a <u>WINDOWPOS</u> data structure that contains information about the window's new size and position.

### Returns

An application should return zero if it processes this message.

### Comments

During this message, modifying any of the values in the WINDOWPOS structure affects the new size, position, or z-order. An application can prevent changes to the window by setting or clearing the appropriate bits in the flags member of the WINDOWPOS structure.

For a window with the WS\_OVERLAPPED or WS\_THICKFRAME style, the DefWindowProc function handles a WM\_WINDOWPOSCHANGING message by sending a WM\_GETMINMAXINFO message to the window. This is done to validate the new size and position of the window and to enforce the CS\_BYTEALIGNCLIENT and CS\_BYTEALIGN client styles. An application can override this functionality by not passing the WM\_WINDOWPOSCHANGING message to the DefWindowProc function.

### See Also

WM\_WINDOWPOSCHANGED, EndDeferWindowPos, SetWindowPos



## WM\_WININICHANGE (2.x)

WM\_WININICHANGE

```
wParam = 0; /* not used, must be zero */  
lParam = (LPARAM) (LPCSTR) pszSection; /* address of string*/
```

An application sends the WM\_WININICHANGE message to all top-level windows after making a change to the Windows initialization file, WIN.INI. The SystemParametersInfo function sends the WM\_WININICHANGE message after an application uses the function to change a setting in the WIN.INI file.

Parameter	Description
pszSection	Value of lParam. Points to a string that specifies the name of the section that has changed (the string does not include the square brackets that enclose the section name).

### Returns

An application should return zero if it processes this message.

### Comments

To send the WM\_WININICHANGE message to all top-level windows, an application can use the SendMessage function with the hwnd parameter set to HWND\_BROADCAST.

If an application changes many different sections in WIN.INI at the same time, the application should send the WM\_WININICHANGE message once with the pszSection parameter set to NULL. Otherwise, an application should send a separate WM\_WININICHANGE message for each change it makes to WIN.INI.

If an application receives a WM\_WININICHANGE message with the pszSection parameter set to NULL, the application should check all sections in WIN.INI that affect the application.

### See Also

SendMessage, SystemParametersInfo

### Messages (3.1)

<u>BM_GETCHECK</u>	Retrieves the check state of a button
<u>BM_GETSTATE</u>	Retrieves the state of a button
<u>BM_SETCHECK</u>	Sets the check state of a button
<u>BM_SETSTATE</u>	Sets the highlight state of a button
<u>BM_SETSTYLE</u>	Changes the style of a button
<u>CB_ADDSTRING</u>	Adds a string to the list box of a combo box
<u>CB_DELETESTRING</u>	Deletes a string in the list box of a combo box
<u>CB_DIR</u>	Adds filenames to the list box of a combo box
<u>CB_FINDSTRING</u>	Finds exact string in the list box of a combo box
<u>CB_FINDSTRINGEXACT</u>	Finds prefix string in the list box of a combo box
<u>CB_GETCOUNT</u>	Gets the number of list-box items in a combo box
<u>CB_GETCURSEL</u>	Gets index of selected list-box item in combo box
<u>CB_GETDROPPEDCONTROLRECT</u>	Gets rectangle of drop-down list box in combo box
<u>CB_GETDROPPEDSTATE</u>	Determines if list box of combo box is visible
<u>CB_GETEDITSEL</u>	Gets position of a selection in an edit control
<u>CB_GETEXTENDEDUI</u>	Determines if combo box has extended interface
<u>CB_GETITEMDATA</u>	Retrieves value associated with combo-box item
<u>CB_GETITEMHEIGHT</u>	Retrieves the height of list items in a combo box
<u>CB_GETLBTEXT</u>	Gets string from the list box of a combo box
<u>CB_GETLBTEXTLEN</u>	Gets length of string in list-box of combo box
<u>CB_INSERTSTRING</u>	Inserts a string into the list box of a combo box
<u>CB_LIMITTEXT</u>	Limits amount of edit-control text in a combo box
<u>CB_RESETCONTENT</u>	Removes all items from the list box of a combo box
<u>CB_SELECTSTRING</u>	Selects matching string in list box of combo box
<u>CB_SETCURSEL</u>	Selects indexed string in list box of combo box
<u>CB_SETEDITSEL</u>	Selects characters in edit control of combo box
<u>CB_SETEXTENDEDUI</u>	Sets the default or extended user interface
<u>CB_SETITEMDATA</u>	Associates a value with combo-box item
<u>CB_SETITEMHEIGHT</u>	Sets the height of list items in a combo box
<u>CB_SHOWDROPDOWN</u>	Shows or hides the list box of a combo box
<u>DM_GETDEFID</u>	Gets the identifier of the default push button
<u>DM_SETDEFID</u>	Sets the default push button of a dialog box
<u>EM_CANUNDO</u>	Determines if edit-control operation can be undone
<u>EM_EMPTYUNDOBUFFER</u>	Resets (clears) undo flag of edit control
<u>EM_FMTLINES</u>	Sets soft line break characters on or off
<u>EM_GETFIRSTVISIBLELINE</u>	Determines topmost line in an edit control
<u>EM_GETHANDLE</u>	Gets handle of memory for multiline edit control
<u>EM_GETLINE</u>	Retrieves line from multiline edit control
<u>EM_GETLINECOUNT</u>	Retrieves number of lines in an MLE
<u>EM_GETMODIFY</u>	Checks whether edit-control contents have changed
<u>EM_GETPASSWORDCHAR</u>	Retrieves edit-control password character
<u>EM_GETRECT</u>	Retrieves coordinates of edit-control rectangle
<u>EM_GETSEL</u>	Gets position of current edit-control selection
<u>EM_GETWORDBREAKPROC</u>	Retrieves the edit-control wordwrap function
<u>EM_LIMITTEXT</u>	Limits the amount of text in an edit control
<u>EM_LINEFROMCHAR</u>	Retrieves a line number from a character index
<u>EM_LINEINDEX</u>	Retrieves character index of edit-control line
<u>EM_LINELENGTH</u>	Retrieves length of line in edit control
<u>EM_LINESCROLL</u>	Scrolls text of a multiline edit control
<u>EM_REPLACESEL</u>	Replaces the current selection in an edit control
<u>EM_SETHANDLE</u>	Sets memory handle for multiline edit control
<u>EM_SETMODIFY</u>	Sets or clears edit-control modification flag
<u>EM_SETPASSWORDCHAR</u>	Sets or removes edit-control password character
<u>EM_SETREADONLY</u>	Sets the read-only state of an edit control
<u>EM_SETRECT</u>	Sets the formatting rectangle of an edit control
<u>EM_SETRECTNP</u>	Sets the formatting rectangle of an edit control
<u>EM_SETSEL</u>	Selects text in a multiline edit control
<u>EM_SETTABSTOPS</u>	Sets tab stops in multiline edit control
<u>EM_SETWORDBREAKPROC</u>	Provides custom word breaks in an edit control
<u>EM_UNDO</u>	Undoes the last edit-control operation
<u>LB_ADDSTRING</u>	Adds a string to a list box
<u>LB_DELETESTRING</u>	Deletes a string in a list box
<u>LB_DIR</u>	Adds a list of filenames to a list box

<u>LB_FINDSTRING</u>	Finds a prefix string in a list box
<u>LB_FINDSTRINGEXACT</u>	Finds an exact string in a list box
<u>LB_GETCARETINDEX</u>	Gets index of list-box item with focus rectangle
<u>LB_GETCOUNT</u>	Retrieves the number of items in a list box
<u>LB_GETCURREL</u>	Retrieves index of selected item in a list box
<u>LB_GETHORIZONTALEXTENT</u>	Retrieves the horizontal extent of a list box
<u>LB_GETITEMDATA</u>	Retrieves the value associated with list-box item
<u>LB_GETITEMHEIGHT</u>	Determines the height of items in a list box
<u>LB_GETITEMRECT</u>	Retrieves the bounding rectangle for an item
<u>LB_GETSEL</u>	Retrieves the selection state of an item
<u>LB_GETSEL COUNT</u>	Retrieves the count of selected list-box items
<u>LB_GETSELITEMS</u>	Lists item numbers of selected list-box items
<u>LB_GETTEXT</u>	Retrieves a string from a list box
<u>LB_GETTEXTLEN</u>	Retrieves the length of a string in a list box
<u>LB_GETTOPINDEX</u>	Retrieves index of first visible list-box item
<u>LB_INSERTSTRING</u>	Inserts a string into a list box
<u>LB_RESETCONTENT</u>	Removes all items from a list box
<u>LB_SELECTSTRING</u>	Selects a matching string in a list box
<u>LB_SELITEMRANGE</u>	Selects consecutive items in a list box
<u>LB_SETCARETINDEX</u>	Sets the focus rectangle in a list box
<u>LB_SETCOLUMNWIDTH</u>	Sets the width of columns in a list box
<u>LB_SETCURSEL</u>	Selects an indexed string in a list box
<u>LB_SETHORIZONTALEXTENT</u>	Sets the horizontal extent of a list box
<u>LB_SETITEMDATA</u>	Associates a value with a list-box item
<u>LB_SETITEMHEIGHT</u>	Sets the height of items in a list box
<u>LB_SETSEL</u>	Selects a string in a multiple-selection list box
<u>LB_SETTABSTOPS</u>	Sets tab stops in a list box
<u>LB_SETTOPINDEX</u>	Ensures that a list-box item is visible
<u>STM_GETICON</u>	Gets icon handle associated with icon resource
<u>STM_SETICON</u>	Associates icon handle with icon resource
<u>WM_ACTIVATE</u>	Indicates a change in the activation state
<u>WM_ACTIVATEAPP</u>	Notifies applications when a new task is activated
<u>WM_ASKCBFORMATNAME</u>	Retrieves the name of the clipboard format
<u>WM_CANCEL MODE</u>	Notifies a window to cancel internal modes
<u>WM_CHANGEFCRCHAIN</u>	Notifies clipboard viewer of removal from chain
<u>WM_CHAR</u>	Passes keyboard events to focus window
<u>WM_CHARTOITEM</u>	Provides list-box keystrokes to owner window
<u>WM_CHILDACTIVATE</u>	Notifies a child window of activation
<u>WM_CHOOSEFONT_GETLOGFONT</u>	Retrieves LOGFONT structure for Font dialog box
<u>WM_CLEAR</u>	Clears an edit control or combo box
<u>WM_CLOSE</u>	Signals a window or application to terminate
<u>WM_COMMAND</u>	Specifies a command message
<u>WM_COMMNOTIFY</u>	Notifies a window about the status of its queues
<u>WM_COMPACTING</u>	Indicates a low memory condition
<u>WM_COMPAREITEM</u>	Determines position of combo-box or list-box item
<u>WM_COPY</u>	Copies a selection to the clipboard
<u>WM_CREATE</u>	Indicates that a window is being created
<u>WM_CTLCOLOR</u>	Indicates that a control is about to be drawn
<u>WM_CUT</u>	Deletes a selection and copies it to the clipboard
<u>WM_DDE_ACK</u>	Acknowledges the receipt of a DDE transaction
<u>WM_DDE_ADVISE</u>	Starts an advise loop with a DDE server
<u>WM_DDE_DATA</u>	Passes a data item to a DDE client
<u>WM_DDE_EXECUTE</u>	Passes a command to a DDE server
<u>WM_DDE_INITIATE</u>	Initiates a DDE conversation
<u>WM_DDE_POKE</u>	Sends an unsolicited data item to a server
<u>WM_DDE_REQUEST</u>	Requests value of a data item from a DDE server
<u>WM_DDE_TERMINATE</u>	Terminates a DDE conversation
<u>WM_DDE_UNADVISE</u>	Ends a DDE advise loop
<u>WM_DEADCHAR</u>	Indicates when a dead key is pressed
<u>WM_DELETEITEM</u>	Indicates owner-drawn item or control is altered
<u>WM_DESTROY</u>	Indicates window is about to be destroyed
<u>WM_DESTROYCLIPBOARD</u>	Notifies owner when clipboard is emptied
<u>WM_DEVMODECHANGE</u>	Indicates when device-mode settings are changed
<u>WM_DRAWCLIPBOARD</u>	Indicates when clipboard contents are changed

<u>WM_DRAWITEM</u>	Indicates when owner-drawn control or menu changes
<u>WM_DROPFILES</u>	Indicates when a file is dropped
<u>WM_ENABLE</u>	Indicates when enable state of window is changing
<u>WM_ENDSESSION</u>	Indicates whether the Windows session is ending
<u>WM_ENTERIDLE</u>	Indicates a modal dialog box or menu is idle
<u>WM_ERASEBKGD</u>	Indicates when background of window needs erasing
<u>WM_FONTCHANGE</u>	Indicates a change in the font-resource pool
<u>WM_GETDLGCODE</u>	Allows processing of control input
<u>WM_GETFONT</u>	Retrieves the font that a control is using
<u>WM_GETMINMAXINFO</u>	Retrieves minimum and maximum sizing information
<u>WM_GETTEXT</u>	Copies the text that corresponds to a window
<u>WM_GETTEXTLENGTH</u>	Determines length of text associated with a window
<u>WM_HSCROLL</u>	Indicates a click in a horizontal scroll bar
<u>WM_HSCROLLCLIPBOARD</u>	Prompts owner to scroll clipboard contents
<u>WM_ICONERASEBKGD</u>	Notifies minimized window to fill icon background
<u>WM_INITDIALOG</u>	Initializes a dialog box
<u>WM_INITMENU</u>	Indicates when a menu is about to become active
<u>WM_INITMENUPOPUP</u>	Indicates when a pop-up menu is being created
<u>WM_KEYDOWN</u>	Indicates when a nonsystem key is pressed
<u>WM_KEYUP</u>	Indicates when a nonsystem key is released
<u>WM_KILLFOCUS</u>	Indicates window is about to lose input focus
<u>WM_LBUTTONDOWNCLK</u>	Indicates double-click of left mouse button
<u>WM_LBUTTONDOWN</u>	Indicates when left mouse button is pressed
<u>WM_LBUTTONUP</u>	Indicates when left mouse button is released
<u>WM_MBUTTONDOWNCLK</u>	Indicates double-click of middle mouse button
<u>WM_MBUTTONDOWN</u>	Indicates when middle mouse button is pressed
<u>WM_MBUTTONUP</u>	Indicates when middle mouse button is released
<u>WM_MDIACTIVATE</u>	Activates a new MDI child window
<u>WM_MDICASCADE</u>	Arranges MDI child windows in a cascade format
<u>WM_MDICREATE</u>	Prompts an MDI client to create a child window
<u>WM_MDIDESTROY</u>	Closes an MDI child window
<u>WM_MDIGETACTIVE</u>	Retrieves data about the active MDI child window
<u>WM_MDIICONARRANGE</u>	Arranges minimized MDI child windows
<u>WM_MDIMAXIMIZE</u>	Maximizes an MDI child window
<u>WM_MDINEXT</u>	Activates the next MDI child window
<u>WM_MDIRESTORE</u>	Prompts an MDI client to restore a child window
<u>WM_MDISETMENU</u>	Replaces the menu of a MDI frame window
<u>WM_MDI TILE</u>	Arranges MDI child windows in a tiled format
<u>WM_MEASUREITEM</u>	Requests dimensions of owner-drawn control
<u>WM_MENUCHAR</u>	Indicates when unknown menu mnemonic is pressed
<u>WM_MENUSELECT</u>	Indicates when a menu item is selected
<u>WM_MOUSEACTIVATE</u>	Indicates a mouse click in an inactive window
<u>WM_MOUSEMOVE</u>	Indicates mouse-cursor movement
<u>WM_MOVE</u>	Indicates the position of a window has changed
<u>WM_NCACTIVATE</u>	Changes the active state of a nonclient area
<u>WM_NCCALCSIZE</u>	Calculates the size of a window's client area
<u>WM_NCCREATE</u>	Indicates a nonclient area is being created
<u>WM_NCDESTROY</u>	Indicates when nonclient area is being destroyed
<u>WM_NCHITTEST</u>	Indicates mouse-cursor movement
<u>WM_NCLBUTTONDOWNCLK</u>	Indicates non-client left button double-click
<u>WM_NCLBUTTONDOWN</u>	Indicates left button pressed in nonclient area
<u>WM_NCLBUTTONUP</u>	Indicates left button released in nonclient area
<u>WM_NCMBUTTONDBLCLK</u>	Indicates middle button nonclient double-click
<u>WM_NCMBUTTONDOWN</u>	Indicates middle button pressed in nonclient area
<u>WM_NCMBUTTONUP</u>	Indicates middle button released in nonclient area
<u>WM_NCMOUSEMOVE</u>	Indicates mouse-cursor movement in nonclient area
<u>WM_NCPAINT</u>	Indicates a window's frame needs painting
<u>WM_NCRBUTTONDOWNCLK</u>	Indicates right button nonclient double-click
<u>WM_NCRBUTTONDOWN</u>	Indicates right button pressed in nonclient area
<u>WM_NCRBUTTONUP</u>	Indicates right button released in nonclient area
<u>WM_NEXTDLGCTL</u>	Sets the focus to a different dialog box control
<u>WM_PAINT</u>	Indicates a window frame needs painting
<u>WM_PAINTCLIPBOARD</u>	Paints the specified portion of the window
<u>WM_PALETTECHANGED</u>	Indicates focus-window has realized its palette

<u>WM_PALETTEISCHANGING</u>	Informs windows about change to palette
<u>WM_PARENTNOTIFY</u>	Notifies parent of child-window activity
<u>WM_PASTE</u>	Inserts clipboard data into an edit control
<u>WM_POWER</u>	Indicates the system is entering suspended mode
<u>WM_QUERYDRAGICON</u>	Requests a cursor handle for a minimized window
<u>WM_QUERYENDSESSION</u>	Requests that the Windows session be ended
<u>WM_QUERYNEWPALETTE</u>	Allows a window to realize its logical palette
<u>WM_QUERYOPEN</u>	Requests that a minimized window be restored
<u>WM_QUEUESYNC</u>	Delimits CBT messages
<u>WM_QUIT</u>	Requests that an application be terminated
<u>WM_RBUTTONDOWNBLCK</u>	Indicates a double-click of right mouse button
<u>WM_RBUTTONDOWN</u>	Indicates when the right mouse button is pressed
<u>WM_RBUTTONUP</u>	Indicates when the right mouse button is released
<u>WM_RENDERALLFORMATS</u>	Notifies owner to render all clipboard formats
<u>WM_RENDERFORMAT</u>	Notifies owner to render particular clipboard data
<u>WM_SETCURSOR</u>	Displays the appropriate mouse cursor shape
<u>WM_SETFOCUS</u>	Indicates when a window has gained input focus
<u>WM_SETFONT</u>	Sets the font for a control
<u>WM_SETREDRAW</u>	Allows or prevents redrawing in a window
<u>WM_SETTEXT</u>	Sets the text of a window
<u>WM_SHOWWINDOW</u>	Indicates a window is about to be hidden or shown
<u>WM_SIZE</u>	Indicates a change in window size
<u>WM_SIZECLIPBOARD</u>	Indicates a change in clipboard size
<u>WM_SPOOLERSTATUS</u>	Indicates when a print job is added or removed
<u>WM_SYSCHAR</u>	Indicates when a System-menu key is pressed
<u>WM_SYSCOLORCHANGE</u>	Indicates when a system color setting is changed
<u>WM_SYSCOMMAND</u>	Indicates when a System-command is requested
<u>WM_SYSDEADCHAR</u>	Indicates when a system dead key is pressed
<u>WM_SYSKEYDOWN</u>	Indicates that ALT plus another key was pressed
<u>WM_SYSKEYUP</u>	Indicates that ALT plus another key was released
<u>WM_SYSTEMERROR</u>	Indicates that a system error has occurred
<u>WM_TIMECHANGE</u>	Indicates that the system time has been set
<u>WM_TIMER</u>	Indicates timeout interval for a timer has elapsed
<u>WM_UNDO</u>	Undoes the last operation in an edit control
<u>WM_USER</u>	Indicates a range of message values
<u>WM_VKEYTOITEM</u>	Provides list-box keystrokes to owner window
<u>WM_VSCROLL</u>	Indicates a click in a vertical scroll bar
<u>WM_VSCROLLCLIPBOARD</u>	Prompts the owner to scroll clipboard contents
<u>WM_WINDOWPOSCHANGED</u>	Notifies a window of a size or position change
<u>WM_WINDOWPOSCHANGING</u>	Notifies a window of a new size or position
<u>WM_WININICHANGE</u>	Notifies applications of change to WIN.INI

## BN\_CLICKED (2.x)

### BN\_CLICKED

The BN\_CLICKED notification message is sent to the parent window when the user clicks a button. Unlike the other button-notification messages, this message is intended for applications written for any version of Windows.

Parameter	Description
wParam	Specifies the control identifier.
lParam	Contains a handle that identifies the button control in its low-order word and the BN_CLICKED notification code in its high-order word.

See Also

[DRAWITEMSTRUCT](#), [WM\\_DRAWITEM](#)

## BN\_DISABLE (2.x)

### BN\_DISABLE

The BN\_DISABLE notification message is sent when a button is disabled. This notification is provided for compatibility with applications written prior to Windows version 3.0. New applications should use the BS\_OWNERDRAW button style and the DRAWITEMSTRUCT structure for this task.

See Also

DRAWITEMSTRUCT, WM\_DRAWITEM

## BN\_DOUBLECLICKED (2.x)

### BN\_DOUBLECLICKED

The BN\_DOUBLECLICKED notification message is sent when the user double clicks a button. This notification is provided for compatibility with applications written prior to Windows version 3.0. New applications should use the BS\_OWNERDRAW button style and the DRAWITEMSTRUCT structure for this task.

See Also

DRAWITEMSTRUCT, WM\_DRAWITEM



## BN\_HILITE (2.x)

### BN\_HILITE

The BN\_HILITE notification message is sent when the user highlights a button. This notification is provided for compatibility with applications written prior to Windows version 3.0. New applications should use the BS\_OWNERDRAW button style and the DRAWITEMSTRUCT structure for this task.

See Also

DRAWITEMSTRUCT, WM\_DRAWITEM

## BN\_PAINT (2.x)

### BN\_PAINT

The BN\_PAINT notification message is sent when a button should be painted. This notification is provided for compatibility with applications written prior to Windows version 3.0. New applications should use the BS\_OWNERDRAW button style and the DRAWITEMSTRUCT structure for this task.

See Also

DRAWITEMSTRUCT, WM\_DRAWITEM

## BN\_UNHILITE (2.x)

### BN\_UNHILITE

The BN\_UNHILITE notification message is sent when the highlight should be removed from a button. This notification is provided for compatibility with applications written prior to Windows version 3.0. New applications should use the BS\_OWNERDRAW button style and the DRAWITEMSTRUCT structure for this task.

See Also

DRAWITEMSTRUCT, WM\_DRAWITEM

## CBN\_CLOSEUP (3.1)

### CBN\_CLOSEUP

The CBN\_CLOSEUP notification message is sent when the list box of a combo box is hidden. The control's parent window receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the combo box.
lParam	Specifies the handle of the combo box in the low-order word, and specifies the CBN_CLOSEUP notification message in the high-order word.

#### Comments

This notification message is not sent to a combo box that has the CBS\_SIMPLE style.

The order in which notifications will be sent cannot be predicted. In particular, a CBN\_SELCHANGE notification may occur either before or after a CBN\_CLOSEUP notification.

#### See Also

CBN\_DROPDOWN, CBN\_SELCHANGE, WM\_COMMAND

## CBN\_DBLCLK (3.0)

### CBN\_DBLCLK

The CBN\_DBLCLK notification message is sent when the user double-clicks a string in the list box of a combo box. The control's parent window receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the combo box.
lParam	Specifies the handle of the combo box in the low-order word and the CBN_DBLCLK notification message in the high-order word.

#### Comments

This notification message can occur only for a combo box with the CBS\_SIMPLE style. For a combo box with the CBS\_DROPDOWN or CBS\_DROPDOWNLIST style, a double-click cannot occur because a single click hides the list box.

#### See Also

CBN\_SELCHANGE, WM\_COMMAND

## CBN\_DROPDOWN (3.0)

### CBN\_DROPDOWN

The CBN\_DROPDOWN notification message is sent when the list box of a combo box is about to be dropped down (made visible). The parent window of the combo box receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the combo box.
lParam	Specifies the handle of the combo box in the low-order word, and specifies the CBN_DROPDOWN notification message in the high-order word.

#### Comments

This notification message can occur only for a combo box with the CBS\_DROPDOWN or CBS\_DROPDOWNLIST style.

#### See Also

CBN\_CLOSEUP, WM\_COMMAND

## CBN\_EDITCHANGE (3.0)

### CBN\_EDITCHANGE

The CBN\_EDITCHANGE notification message is sent after the user has taken an action that may have altered the text in the edit-control portion of a combo box. Unlike the CBN\_EDITUPDATE notification message, this notification message is sent after Windows updates the screen. The parent window of the combo box receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the combo box.
lParam	Specifies the handle of the combo box in the low-order word, and specifies the CBN_EDITCHANGE notification message in the high-order word.

#### Comments

This message does not occur if the combo box has the CBS\_DROPDOWNLIST style.

#### See Also

CBN\_EDITUPDATE, WM\_COMMAND

## CBN\_EDITUPDATE (3.0)

### CBN\_EDITUPDATE

The CBN\_EDITUPDATE notification message is sent when the edit-control portion of a combo box is about to display altered text. This notification is sent after the control has formatted the text, but before it displays the text. The parent window of the combo box receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the combo box.
lParam	Specifies the handle of the combo box in the low-order word, and specifies the CBN_EDITUPDATE notification message in the high-order word.

#### Comments

This message does not occur if the combo box has the CBS\_DROPDOWNLIST style.

#### See Also

CBN\_EDITCHANGE, WM\_COMMAND



## CBN\_ERRSPACE (3.0)

### CBN\_ERRSPACE

The CBN\_ERRSPACE notification message is sent when a combo box cannot allocate enough memory to meet a specific request. The parent window of the combo box receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the combo box.
lParam	Specifies the handle of the combo box in the low-order word, and specifies the CBN_ERRSPACE notification message in the high-order word.

See Also

WM\_COMMAND

## CBN\_KILLFOCUS (3.0)

### CBN\_KILLFOCUS

The CBN\_KILLFOCUS notification message is sent when a combo box loses the input focus. The parent window of the combo box receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the combo box.
lParam	Specifies the handle of the combo box in the low-order word, and specifies the CBN_KILLFOCUS notification message in the high-order word.

See Also

CBN\_SETFOCUS, WM\_COMMAND

## CBN\_SELCHANGE (3.0)

### CBN\_SELCHANGE

The CBN\_SELCHANGE notification message is sent when the selection in the list box of a combo box is about to be changed as a result of the user either clicking in the list box or changing the selection by using the arrow keys. The parent window of the combo box receives this code through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the combo box.
lParam	Specifies the handle of the combo box in the low-order word, and specifies the CBN_SELCHANGE notification message in the high-order word.

See Also

CBN\_DBLCLK, CB\_SETCURSEL, WM\_COMMAND

## CBN\_SELENDNCANCEL (3.1)

### CBN\_SELENDNCANCEL

The CBN\_SELENDNCANCEL notification message is sent when the user clicks an item and then clicks another window or control to hide the list box of a combo box. This notification message is sent before the CBN\_CLOSEUP notification message to indicate that the user's selection should be ignored.

Parameter	Description
wParam	Specifies the identifier of the combo box.
lParam	Specifies the handle of the combo box in the low-order word, and specifies the CBN_SELENDNCANCEL notification message in the high-order word.

#### Comments

The CBN\_SELENDNCANCEL or CBN\_SELENDOK notification message is sent even if the CBN\_CLOSEUP notification message is not sent (as in the case of a combo box with the CBS\_SIMPLE style).

#### See Also

CBN\_SELENDOK, WM\_COMMAND

## CBN\_SELENDOK (3.1)

### CBN\_SELENDOK

The CBN\_SELENDOK notification message is sent when the user selects an item and then either presses the ENTER key or clicks the DOWN ARROW key to hide the list box of a combo box. This notification message is sent before the CBN\_CLOSEUP notification message to indicate that the user's selection should be considered valid.

Parameter	Description
wParam	Specifies the identifier of the combo box.
lParam	Specifies the handle of the combo box in the low-order word, and specifies the CBN_SELENDOK notification message in the high-order word.

#### Comments

The CBN\_SELENDOK or CBN\_SELENDNCANCEL notification message is sent even if the CBN\_CLOSEUP notification message is not sent (as in the case of a combo box with the CBS\_SIMPLE style).

#### See Also

CBN\_SELENDNCANCEL, WM\_COMMAND

## CBN\_SETFOCUS (3.0)

### CBN\_SETFOCUS

The CBN\_SETFOCUS notification message is sent when a combo box receives the input focus. The parent window of the combo box receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the combo box.
lParam	Specifies the handle of the combo box in the low-order word, and specifies the CBN_SETFOCUS notification message in the high-order word.

See Also

CBN\_KILLFOCUS, WM\_COMMAND

## EN\_CHANGE (2.x)

### EN\_CHANGE

The EN\_CHANGE notification message is sent when the user has taken an action that may have altered text in an edit control. Unlike the EN\_UPDATE notification message, this notification message is sent after Windows updates the display. The control's parent window receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the edit control.
lParam	Specifies the handle of the edit control in the low-order word, and specifies the EN_CHANGE notification message in the high-order word.

See Also

EN\_UPDATE, WM\_COMMAND

## EN\_ERRSPACE (2.x)

### EN\_ERRSPACE

The EN\_ERRSPACE notification message is sent when an edit control cannot allocate enough memory to meet a specific request. The control's parent window receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the edit control.
lParam	Specifies the handle of the edit control in the low-order word, and specifies the EN_ERRSPACE notification message in the high-order word.

See Also

WM\_COMMAND



## EN\_HSCROLL (2.x)

### EN\_HSCROLL

The EN\_HSCROLL notification message is sent when the user clicks an edit control's horizontal scroll bar. The control's parent window receives this notification message through a WM\_COMMAND message. The parent window is notified before the screen is updated.

Parameter	Description
wParam	Specifies the identifier of the edit control.
lParam	Specifies the handle of the edit control in the low-order word, and specifies the EN_HSCROLL notification message in the high-order word.

See Also

EN\_VSCROLL, WM\_COMMAND

## EN\_KILLFOCUS (2.x)

### EN\_KILLFOCUS

The EN\_KILLFOCUS notification message is sent when an edit control loses the input focus. The control's parent window receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the edit control.
lParam	Specifies the handle of the edit control in the low-order word, and specifies the EN_KILLFOCUS notification message in the high-order word.

See Also

EN\_SETFOCUS, WM\_COMMAND

## EN\_MAXTEXT (3.0)

### EN\_MAXTEXT

The EN\_MAXTEXT notification message is sent when the current insertion has exceeded the specified number of characters for the edit control. The insertion has been truncated.

This message is also sent when an edit control does not have the ES\_AUTOHSCROLL style and the number of characters to be inserted would exceed the width of the edit control.

This message is also sent when an edit control does not have the ES\_AUTOVSCROLL style and the total number of lines resulting from a text insertion would exceed the height of the edit control.

The control's parent window receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the edit control.
lParam	Specifies the handle of the edit control in the low-order word, and specifies the EN_MAXTEXT notification message in the high-order word.

See Also

EM\_LIMITTEXT, WM\_COMMAND

## EN\_SETFOCUS (2.x)

### EN\_SETFOCUS

The EN\_SETFOCUS notification message is sent when an edit control receives the input focus. The control's parent window receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the edit control.
lParam	Specifies the handle of the edit control in the low-order word, and specifies the EN_SETFOCUS notification message in the high-order word.

See Also

EN\_KILLFOCUS, WM\_COMMAND

## EN\_UPDATE (2.x)

### EN\_UPDATE

The EN\_UPDATE notification message is sent when an edit control is about to display altered text. This notification is sent after the control has formatted the text but before it screens the text. This makes it possible to alter the window size, if necessary. The control's parent window receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the edit control.
lParam	Specifies the handle of the edit control in the low-order word, and specifies the EN_UPDATE notification message in the high-order word.

See Also

EN\_CHANGE, WM\_COMMAND

## EN\_VSCROLL (2.x)

### EN\_VSCROLL

The EN\_VSCROLL notification message is sent when the user clicks an edit control's vertical scroll bar. The control's parent window receives this notification message through a WM\_COMMAND message. The parent window is notified before the screen is updated.

Parameter	Description
wParam	Specifies the identifier of the edit control.
lParam	Specifies the handle of the edit control in the low-order word, and specifies the EN_VSCROLL notification message in the high-order word.

See Also

EN\_HSCROLL, WM\_COMMAND

## LBN\_DBLCLK (2.x)

### LBN\_DBLCLK

The LBN\_DBLCLK notification message is sent when the user double-clicks a string in a list box. The parent window of the list box receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the list box.
lParam	Specifies the handle of the list box in the low-order word, and specifies the LBN_DBLCLK notification message in the high-order word.

#### Comments

Only a list box that has LBS\_NOTIFY style will send this notification message.

#### See Also

LBN\_SELCHANGE, WM\_COMMAND

## LBN\_ERRSPACE (2.x)

### LBN\_ERRSPACE

The LBN\_ERRSPACE notification message is sent when a list box cannot allocate enough memory to meet a specific request. The parent window of the list box receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the list box.
lParam	Specifies the handle of the list box in the low-order word, and specifies the LBN_ERRSPACE notification message in the high-order word.

See Also

WM\_COMMAND



## LBN\_KILLFOCUS (3.0)

### LBN\_KILLFOCUS

The LBN\_KILLFOCUS notification message is sent when a list box loses the input focus. The parent window of the list box receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the list box.
lParam	Specifies the handle of the list box in the low-order word, and specifies the LBN_KILLFOCUS notification message in the high-order word.

See Also

LBN\_SETFOCUS, WM\_COMMAND

## LBN\_SELCANCEL (3.1)

### LBN\_SELCANCEL

The LBN\_SELCANCEL notification message is sent when the user cancels the selection in a list box. The parent window of the list box receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the list box.
lParam	Specifies the handle of the list box in the low-order word, and specifies the LBN_SELCANCEL notification message in the high-order word.

#### Comments

This notification applies only to a list box that has the LBS\_NOTIFY style.

#### See Also

LBN\_DBLCLK, LBN\_SELCHANGE, LB\_SETCURSEL, WM\_COMMAND

## LBN\_SELCHANGE (2.x)

### LBN\_SELCHANGE

The LBN\_SELCHANGE notification message is sent when the selection in a list box is about to change. The parent window of the list box receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the list box.
lParam	Specifies the handle of the list box in the low-order word, and specifies the LBN_SELCHANGE notification message in the high-order word.

#### Comments

This notification is not sent if the selection is changed by the LB\_SETCURSEL message.

This notification applies only to a list box that has the LBS\_NOTIFY style.

The LBN\_SELCHANGE notification is sent for a multiple-selection list box whenever the user presses an arrow key, even if the selection does not change.

#### See Also

LBN\_DBLCLK, LBN\_SELCANCEL, LB\_SETCURSEL, WM\_COMMAND

## LBN\_SETFOCUS (3.0)

### LBN\_SETFOCUS

The LBN\_SETFOCUS notification message is sent when a list box receives the input focus. The parent window of the list box receives this notification message through a WM\_COMMAND message.

Parameter	Description
wParam	Specifies the identifier of the list box.
lParam	Specifies the handle of the list box in the low-order word, and specifies the LBN_SETFOCUS notification message in the high-order word.

See Also

LBN\_KILLFOCUS, WM\_COMMAND

### Notification messages (3.1)

<u>BN_CLICKED</u>	Indicates the user clicked a button
<u>BN_DISABLE</u>	Indicates a button is disabled
<u>BN_DOUBLECLICKED</u>	Indicates the user double-clicked a button
<u>BN_HILITE</u>	Indicates the user highlighted a button
<u>BN_PAINT</u>	Indicates the button should be painted
<u>BN_UNHILITE</u>	Indicates the highlight should be removed
<u>CBN_CLOSEUP</u>	Indicates the list box of a combo box has closed
<u>CBN_DBLCLK</u>	Indicates the user double-clicked a string
<u>CBN_DROPDOWN</u>	Indicates the list box of a combo box is dropping down
<u>CBN_EDITCHANGE</u>	Indicates the user has changed text in the edit control
<u>CBN_EDITUPDATE</u>	Indicates altered text is about to be displayed
<u>CBN_ERRSPACE</u>	Indicates the combo box is out of memory
<u>CBN_KILLEFOCUS</u>	Indicates the combo box is losing the input focus
<u>CBN_SELCHANGE</u>	Indicates a new combo box list item is selected
<u>CBN_SELENDCANCEL</u>	Indicates the user's selection should be cancelled
<u>CBN_SELENDOK</u>	Indicates the user's selection is valid
<u>CBN_SETFOCUS</u>	Indicates the combo box is receiving the input focus
<u>EN_CHANGE</u>	Indicates the display is updated after text changes
<u>EN_ERRSPACE</u>	Indicates the edit control is out of memory
<u>EN_HSCROLL</u>	Indicates the user clicked the scroll bar
<u>EN_KILLEFOCUS</u>	Indicates the edit control is losing the input focus
<u>EN_MAXTEXT</u>	Indicates the insertion is truncated
<u>EN_SETFOCUS</u>	Indicates the edit-control is receiving the input focus
<u>EN_UPDATE</u>	Indicates edit-control is about to display altered text
<u>EN_VSCROLL</u>	Indicates the user clicked the vertical scroll bar
<u>LBN_DBLCLK</u>	Indicates that the user double-clicked a string
<u>LBN_ERRSPACE</u>	Indicates the list box is out of memory
<u>LBN_KILLEFOCUS</u>	Indicates the list box is losing the input focus
<u>LBN_SELCANCEL</u>	Indicates the selection is cancelled
<u>LBN_SELCHANGE</u>	Indicates the selection is about to change
<u>LBN_SETFOCUS</u>	Indicates the list box is receiving the input focus

## OleActivate (3.1)

```
#include ole.h

OLESTATUS OleActivate(lpObject, verb, fShow, fTakeFocus, hwnd, lprcBound)
LPOLEOBJECT lpObject;          /* address of object to activate */
UINT verb;                      /* operation to perform */
BOOL fShow;                     /* whether to show window */
BOOL fTakeFocus;               /* whether server gets focus */
HWND hwnd;                     /* window handle of destination document */
const RECT FAR* lprcBound;     /* bounding rectangle for object display */
```

The OleActivate function opens an object for an operation. Typically, the object is edited or played.

Parameter	Description
lpObject	Points to the object to activate.
verb	Specifies which operation to perform (0 = the primary verb, 1 = the secondary verb, and so on).
fShow	Specifies whether the window is to be shown. If the window is to be shown, this value is TRUE; otherwise, it is FALSE.
fTakeFocus	Specifies whether the server should get the focus. If the server should get the focus, this value is TRUE; otherwise, it is FALSE. This parameter is relevant only if the fShow parameter is TRUE.
hwnd	Identifies the window of the document containing the object.
lprcBound	Points to a RECT structure containing the coordinates of the bounding rectangle in which the destination document displays the object. The mapping mode of the device context determines the units for these coordinates.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_BUSY  
OLE\_ERROR\_OBJECT  
OLE\_WAIT\_FOR\_RELEASE

### Comments

Typically, a server is launched in a separate window; editing then occurs asynchronously. The client is notified of changes to the object through the callback function.

A client application might set the fShow parameter to FALSE if a server needed to remain active without being visible on the display. (In this case, the application would also use the OleSetData function.)

Client applications typically specify the primary verb when the user double-clicks an object. The server can take any action in response to the specified verb. If the server supports only one action, it takes that action no matter which value is passed in the verb parameter.

In future releases of the object linking and embedding (OLE) protocol, the hwnd and lprcBound parameters will be used to help determine the placement of the server's editing window.

### See Also

OleQueryOpen, OleSetData, RECT

## OleBlockServer (3.1)

#include ole.h

```
OLESTATUS OleBlockServer(lhSrvr)
LHSERVER lhSrvr;          /* handle of server */
```

The OleBlockServer function causes requests to the server to be queued until the server calls the OleUnblockServer function.

Parameter	Description
lhSrvr	Identifies the server for which requests are to be queued.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be OLE\_ERROR\_HANDLE.

### Comments

The server must call the OleUnblockServer function after calling the OleBlockServer function.

A server application can use the OleBlockServer and OleUnblockServer functions to control when the server library processes requests from client applications. Because only messages from the client to the server are blocked, a blocked server can continue to send messages to client applications.

A server application receives a handle when it calls the OleRegisterServer function.

### See Also

OleRegisterServer, OleUnblockServer

### OleClone (3.1)

```
#include ole.h

OLESTATUS OleClone(lpObject, lpClient, lhClientDoc, lpszObjname, lpObject)
LPOLEOBJECT lpObject; /* address of object to copy */
LPOLECLIENT lpClient;
/* address of OLECLIENT for new object
/
LHCLIENTDOC lhClientDoc;
/* long handle of client document
/
LPCSTR lpszObjname;
/* address of string for object name
/
LPOLEOBJECT FAR* lpObject;
/* address of pointer to new object
/
```

The OleClone function makes a copy of an object. The copy is identical to the source object, but it is not connected to the server.

Parameter	Description
lpObject	Points to the object to copy.
lpClient	Points to an <u>OLECLIENT</u> structure for the new object.
lhClientDoc	Identifies the client document in which the object is to be created.
lpszObjname	Points to a null-terminated string specifying the client's name for the object. This name must be unique with respect to the names of any other objects in the document and cannot contain a slash mark (/).
lpObject	Points to a variable where the library will store the long pointer to the new object.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_BUSY  
OLE\_ERROR\_HANDLE  
OLE\_ERROR\_OBJECT  
OLE\_WAIT\_FOR\_RELEASE

#### Comments

Client applications often use the OleClone function to support the Undo command.

A client application can supply a new OLECLIENT structure for the cloned object, if required.

#### See Also

OleEqual, OLECLIENT



### OleClose (3.1)

```
#include ole.h
```

```
OLESTATUS OleClose(lpObject)  
LPOLEOBJECT lpObject; /* address of object to close */
```

The OleClose function closes the specified open object. Closing an object terminates the connection with the server application.

Parameter	Description
lpObject	Points to the object to close.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_BUSY  
OLE_ERROR_OBJECT  
OLE_WAIT_FOR_RELEASE
```

#### See Also

[OleActivate](#), [OleDelete](#), [OleReconnect](#)

### OleCopyFromLink (3.1)

```
#include ole.h

OLESTATUS OleCopyFromLink(lpObject, lpszProtocol, lpClient, lhClientDoc, lpszObjname,
                          lpObject)
LPOLEOBJECT lpObject;          /* address of object to embed */
LPCSTR lpszProtocol;          /* address of protocol name */
LPOLECLIENT lpClient;        /* address of client structure */
LHCLIENTDOC lhClientDoc;     /* long handle of client document */
LPCSTR lpszObjname;          /* address of string for object name */
LPOLEOBJECT FAR* lpObject;    /* address of pointer to new object */
```

The OleCopyFromLink function makes an embedded copy of a linked object.

Parameter	Description
lpObject	Points to the linked object that is to be embedded.
lpszProtocol	Points to a null-terminated string specifying the name of the protocol required for the new embedded object. Currently, this value can be StdFileEditing (the name of the object linking and embedding protocol).
lpClient	Points to an <u>OLECLIENT</u> structure for the new object.
lhClientDoc	Identifies the client document in which the object is to be created.
lpszObjname	Points to a null-terminated string specifying the client's name for the object.
lpObject	Points to a variable where the library will store the long pointer to the new object.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_BUSY  
OLE\_ERROR\_HANDLE  
OLE\_ERROR\_NAME  
OLE\_ERROR\_OBJECT  
OLE\_ERROR\_PROTOCOL  
OLE\_WAIT\_FOR\_RELEASE

#### Comments

Making an embedded copy of a linked object may involve starting the server application.

#### See Also

OleObjectConvert

## OleCopyToClipboard (3.1)

#include ole.h

```
OLESTATUS OleCopyToClipboard(lpObject)
LPOLEOBJECT lpObject;      /* address of object */
```

The OleCopyToClipboard function puts the specified object on the clipboard.

Parameter	Description
lpObject	Points to the object to copy to the clipboard.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be OLE\_ERROR\_OBJECT.

### Comments

A client application typically calls the OleCopyToClipboard function when a user chooses the Copy or Cut command from the Edit menu.

The client application should open and empty the clipboard, call the OleCopyToClipboard function, and close the clipboard.

### OleCreate (3.1)

```
#include ole.h

OLESTATUS OleCreate(lpszProtocol, lpClient, lpszClass, lhClientDoc, lpszObjname, lpIpObject,
                    renderopt, cfFormat)
LPCSTR lpszProtocol; /* address of string for protocol name */
LPOLECLIENT lpClient; /* address of client structure */
LPCSTR lpszClass; /* address of string for classname */
LHCLIENTDOC lhClientDoc; /* long handle of client document */
LPCSTR lpszObjname; /* address of string for object name */
LPOLEOBJECT FAR* lpIpObject; /* address of pointer to object */
OLEOPT_RENDER renderopt; /* rendering options */
OLECLIPFORMAT cfFormat; /* clipboard format
```

The OleCreate function creates an embedded object of a specified class. The server is opened to perform the initial editing.

Parameter	Description								
lpszProtocol	Points to a null-terminated string specifying the name of the protocol required for the new embedded object. Currently, this value can be StdFileEditing (the name of the object linking and embedding protocol).								
lpClient	Points to an <u>OLECLIENT</u> structure for the new object.								
lpszClass	Points to a null-terminated string specifying the registered name of the class of the object to be created.								
lhClientDoc	Identifies the client document in which the object is to be created.								
lpszObjname	Points to a null-terminated string specifying the client's name for the object. This name must be unique with respect to the names of any other objects in the document and cannot contain a slash mark (/).								
lpIpObject	Points to a variable where the library will store the long pointer to the new object.								
renderopt	Specifies the client's preference for presentation data for the object. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>olerender_draw</td><td>The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.</td></tr><tr><td>olerender_format</td><td>The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.</td></tr><tr><td>olerender_none</td><td>The client library does not obtain any presentation data and does not draw the object.</td></tr></tbody></table>	Value	Meaning	olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.	olerender_format	The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.	olerender_none	The client library does not obtain any presentation data and does not draw the object.
Value	Meaning								
olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.								
olerender_format	The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.								
olerender_none	The client library does not obtain any presentation data and does not draw the object.								

#### Returns

cfFormat Specifies the clipboard format when the renderopt parameter is olerender\_format. This clipboard format is used in a subsequent call to OleGetData. If this clipboard format is CF\_METAFILEPICT, CF\_DIB, or CF\_BITMAP, the library manages the data and draws the object. The library does not support drawing for any other formats. The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_HANDLE  
OLE\_ERROR\_NAME  
OLE\_ERROR\_PROTOCOL  
OLE\_WAIT\_FOR\_RELEASE

#### Comments

The `olerender_none` rendering option is typically used to support hyperlinks. With this option, the client does not call OleDraw and calls OleGetData only for ObjectLink, OwnerLink, and Native formats.

The `olerender_format` rendering option allows a client to compute data (instead of painting it), use an unusual data format, or modify a standard data format. With this option, the client does not call OleDraw. The client calls OleGetData to retrieve data in the specified format.

The `olerender_draw` rendering option is the most typical option. It is the easiest rendering option for the client to implement (the client simply calls OleDraw), and it allows the most flexibility. An object handler can exploit this flexibility to store no presentation data, a private presentation data format, or several different formats that it can choose among dynamically. Future implementations of object linking and embedding (OLE) may also exploit the flexibility that is inherent in this option.

#### See Also

OleCreateFromClip, OleCreateFromTemplate, OleDraw, OleGetData

### OleCreateFromClip (3.1)

```
#include ole.h

OLESTATUS OleCreateFromClip(lpszProtocol, lpClient, lhClientDoc, lpszObjname, lpIpObject,
                           renderopt, cfFormat)
LPCSTR lpszProtocol;          /* address of string for protocol name */
LPOLECLIENT lpClient;       /* address of client structure */
LHCLIENTDOC lhClientDoc;    /* long handle of client document */
LPCSTR lpszObjname;         /* address of string for object name */
LPOLEOBJECT FAR* lpIpObject; /* address of pointer to object */
OLEOPT_RENDER renderopt;    /* rendering options */
OLECLIPFORMAT cfFormat;     /* clipboard format */
```

The OleCreateFromClip function creates an object from the clipboard.

Parameter	Description								
lpszProtocol	Points to a null-terminated string specifying the name of the protocol required for the new embedded object. Currently, this value can be StdFileEditing (the name of the object linking and embedding protocol) or Static (for uneditable pictures only).								
lpClient	Points to an <u>OLECLIENT</u> structure allocated and initialized by the client application. This pointer is used to locate the callback function and is passed in callback notifications.								
lhClientDoc	Identifies the client document in which the object is being created.								
lpszObjname	Points to a null-terminated string specifying the client's name for the object. This name must be unique with respect to the names of any other objects in the document and cannot contain a slash mark (/).								
lpIpObject	Points to a variable where the library will store the long pointer to the new object.								
renderopt	Specifies the client's preference for presentation data for the object. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>olerender_draw</td><td>The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.</td></tr><tr><td>olerender_format</td><td>The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.</td></tr><tr><td>olerender_none</td><td>The client library does not obtain any presentation data and does not draw the object.</td></tr></tbody></table>	Value	Meaning	olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.	olerender_format	The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.	olerender_none	The client library does not obtain any presentation data and does not draw the object.
Value	Meaning								
olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.								
olerender_format	The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.								
olerender_none	The client library does not obtain any presentation data and does not draw the object.								
cfFormat	Specifies the clipboard format when the renderopt parameter is olerender_format. This clipboard format is used in a subsequent call to <u>OleGetData</u> . If this clipboard format is CF_METAFILEPICT, CF_DIB, or CF_BITMAP, the library manages the data and draws the object. The library does not support drawing for any other formats.								

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_CLIP  
OLE\_ERROR\_FORMAT

OLE\_ERROR\_HANDLE  
OLE\_ERROR\_NAME  
OLE\_ERROR\_OPTION  
OLE\_ERROR\_PROTOCOL  
OLE\_WAIT\_FOR\_RELEASE

#### Comments

The client application should open and empty the clipboard, call the `OleCreateFromClip` function, and close the clipboard.

The `olerender_none` rendering option is typically used to support hyperlinks. With this option, the client does not call `OleDraw` and calls `OleGetData` only for ObjectLink, OwnerLink, and Native formats.

The `olerender_format` rendering option allows a client to compute data (instead of painting it), use an unusual data format, or modify a standard data format. With this option, the client does not call `OleDraw`. The client calls `OleGetData` to retrieve data in the specified format.

The `olerender_draw` rendering option is the most typical option. It is the easiest rendering option for the client to implement (the client simply calls `OleDraw`), and it allows the most flexibility. An object handler can exploit this flexibility to store no presentation data, a private presentation data format, or several different formats that it can choose among dynamically. Future implementations of object linking and embedding (OLE) may also exploit the flexibility that is inherent in this option.

#### See Also

[OleCreate](#), [OleCreateFromTemplate](#), [OleDraw](#), [OleGetData](#), [OleQueryCreateFromClip](#)

### OleCreateFromFile (3.1)

```
#include ole.h

OLESTATUS OleCreateFromFile(lpszProtocol, lpClient, lpszClass, lpszFile, lhClientDoc, lpszObjname,
                           lpObject, renderopt, cfFormat)
LPCSTR lpszProtocol;          /* address of string for protocol name */
LPOLECLIENT lpClient;        /* address of client structure */
LPCSTR lpszClass;            /* address of string for class name */
LPCSTR lpszFile;             /* address of string for filename */
LHCLIENTDOC lhClientDoc;     /* long handle of client document */
LPCSTR lpszObjname;          /* address of string for object name */
LPOLEOBJECT FAR* lpObject;   /* address of pointer to object */
OLEOPT_RENDER renderopt;     /* rendering options */
OLECLIPFORMAT cfFormat;     /* clipboard format
```

The OleCreateFromFile function creates an embedded object from the contents of a named file.

Parameter	Description								
lpszProtocol	Points to a null-terminated string specifying the name of the protocol required for the new embedded object. Currently, this value can be StdFileEditing (the name of the object linking and embedding protocol).								
lpClient	Points to an <u>OLECLIENT</u> structure allocated and initialized by the client application. This pointer is used to locate the callback function and is passed in callback notifications.								
lpszClass	Points to a null-terminated string specifying the name of the class for the new object. If this value is NULL, the library uses the extension of the filename pointed to by the lpszFile parameter to find the class name for the object.								
lpszFile	Points to a null-terminated string specifying the name of the file containing the object.								
lhClientDoc	Identifies the client document in which the object is being created.								
lpszObjname	Points to a null-terminated string specifying the client's name for the object. This name must be unique with respect to the names of any other objects in the document and cannot contain a slash mark (/).								
lpObject	Points to a variable where the library will store the long pointer to the new object.								
renderopt	Specifies the client's preference for presentation data for the object. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>olerender_draw</td><td>The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.</td></tr><tr><td>olerender_format</td><td>The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.</td></tr><tr><td>olerender_none</td><td>The client library does not obtain any presentation data and does not draw the object.</td></tr></tbody></table>	Value	Meaning	olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.	olerender_format	The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.	olerender_none	The client library does not obtain any presentation data and does not draw the object.
Value	Meaning								
olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.								
olerender_format	The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.								
olerender_none	The client library does not obtain any presentation data and does not draw the object.								



**cfFormat** Specifies the clipboard format when the `renderopt` parameter is `olerender_format`. This clipboard format is used in a subsequent call to `OleGetData`. If this clipboard format is `CF_METAFILEPICT`, `CF_DIB`, or `CF_BITMAP`, the library manages the data and draws the object. The library does not support drawing for any other formats.

#### Returns

The return value is `OLE_OK` if the function is successful. Otherwise, it is an error value, which may be one of the following:

`OLE_ERROR_CLASS`  
`OLE_ERROR_HANDLE`  
`OLE_ERROR_MEMORY`  
`OLE_ERROR_NAME`  
`OLE_ERROR_PROTOCOL`  
`OLE_WAIT_FOR_RELEASE`

#### Comments

When a client application calls the `OleCreateFromFile` function, the server is started to render the Native and presentation data and then is closed. (If the server and document are already open, this function simply retrieves the information, without closing the server.) The server does not show the object to the user for editing.

The `olerender_none` rendering option is typically used to support hyperlinks. With this option, the client does not call `OleDraw` and calls `OleGetData` only for `ObjectLink`, `OwnerLink`, and `Native` formats.

The `olerender_format` rendering option allows a client to compute data (instead of painting it), use an unusual data format, or modify a standard data format. With this option, the client does not call `OleDraw`. The client calls `OleGetData` to retrieve data in the specified format.

The `olerender_draw` rendering option is the most typical option. It is the easiest rendering option for the client to implement (the client simply calls `OleDraw`), and it allows the most flexibility. An object handler can exploit this flexibility to store no presentation data, a private presentation data format, or several different formats that it can choose among dynamically. Future implementations of object linking and embedding (OLE) may also exploit the flexibility that is inherent in this option.

If a client application accepts files dropped from File Manager, it should respond to the `WM_DROPFILES` message by calling `OleCreateFromFile` and specifying `Packager` for the `lpszClass` parameter to indicate Microsoft Windows Object Packager.

#### See Also

[OleCreate](#), [OleCreateFromTemplate](#), [OleDraw](#), [OleGetData](#)

### OleCreateFromTemplate (3.1)

```
#include ole.h

OLESTATUS OleCreateFromTemplate(lpszProtocol, lpClient, lpszTemplate, lhClientDoc,
                               lpszObjname, lp lpObject, renderopt, cfFormat)
LPCSTR lpszProtocol;          /* address of string for protocol name */
LPOLECLIENT lpClient;        /* address of client structure */
LPCSTR lpszTemplate;         /* address of string for path of file */
LHCLIENTDOC lhClientDoc;    /* long handle of client document */
LPCSTR lpszObjname;         /* address of string for object name */
LPOLEOBJECT FAR* lp lpObject; /* address of pointer to object */
OLEOPT_RENDER renderopt;    /* rendering options */
OLECLIPFORMAT cfFormat;     /* clipboard format */
```

The OleCreateFromTemplate function creates an object by using another object as a template. The server is opened to perform the initial editing.

Parameter	Description								
lpszProtocol	Points to a null-terminated string specifying the name of the protocol required for the new embedded object. Currently, this value can be StdFileEditing (the name of the object linking and embedding protocol).								
lpClient	Points to an <u>OLECLIENT</u> structure for the new object.								
lpszTemplate	Points to a null-terminated string specifying the path of the file to be used as a template for the new object. The server is opened for editing and loads the initial state of the new object from the named template file.								
lhClientDoc	Identifies the client document in which the object is being created.								
lpszObjname	Points to a null-terminated string specifying the client's name for the object. This name must be unique with respect to the names of any other objects in the document and cannot contain a slash mark (/).								
lp lpObject	Points to a variable where the library will store the long pointer to the new object.								
renderopt	Specifies the client's preference for presentation data for the object. This parameter can be one of the following values: <table border="1" data-bbox="462 1480 1399 1743"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>olerender_draw</td> <td>The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.</td> </tr> <tr> <td>olerender_format</td> <td>The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.</td> </tr> <tr> <td>olerender_none</td> <td>The client library does not obtain any presentation data and does not draw the object.</td> </tr> </tbody> </table>	Value	Meaning	olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.	olerender_format	The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.	olerender_none	The client library does not obtain any presentation data and does not draw the object.
Value	Meaning								
olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.								
olerender_format	The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.								
olerender_none	The client library does not obtain any presentation data and does not draw the object.								
cfFormat	Specifies the clipboard format when the renderopt parameter is olerender_format. This clipboard format is used in a subsequent call to the <u>OleGetData</u> function. If this clipboard format is CF_METAFILEPICT, CF_DIB, or CF_BITMAP, the library manages the data and draws the object. The library does not support drawing for any other formats.								

## Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_CLASS  
OLE\_ERROR\_HANDLE  
OLE\_ERROR\_MEMORY  
OLE\_ERROR\_NAME  
OLE\_ERROR\_PROTOCOL  
OLE\_WAIT\_FOR\_RELEASE

## Comments

The client library uses the filename extension of the file specified in the lpszTemplate parameter to identify the server for the object. The association between the extension and the server is stored in the registration database.

The olerender\_none rendering option is typically used to support hyperlinks. With this option, the client does not call OleDraw and calls OleGetData only for ObjectLink, OwnerLink, and Native formats.

The olerender\_format rendering option allows a client to compute data (instead of painting it), use an unusual data format, or modify a standard data format. With this option, the client does not call OleDraw. The client calls OleGetData to retrieve data in the specified format.

The olerender\_draw rendering option is the most typical option. It is the easiest rendering option for the client to implement (the client simply calls OleDraw), and it allows the most flexibility. An object handler can exploit this flexibility to store no presentation data, a private presentation data format, or several different formats that it can choose among dynamically. Future implementations of object linking and embedding (OLE) may also exploit the flexibility that is inherent in this option.

## See Also

OleCreate, OleCreateFromClip, OleDraw, OleGetData, OleObjectConvert

### OleCreateInvisible (3.1)

```
#include ole.h

OLESTATUS OleCreateInvisible(lpszProtocol, lpClient, lpszClass, lhClientDoc, lpszObjname,
                             lplpObject, renderopt, cfFormat, fActivate)
/* address of string for protocol name */
LPCSTR lpszProtocol;
LPOLECLIENT lpClient;
/* address of client structure
/
LPCSTR lpszClass;
/* address of string for classname
/
LHCLIENTDOC lhClientDoc;
/* long handle of client document
/
LPCSTR lpszObjname;
/* address of string for object name
/
LPOLEOBJECT FAR* lplpObject;
/* address of pointer to object
/
OLEOPT_RENDER renderopt;
/* rendering options
/
OLECLIPFORMAT cfFormat;
/* clipboard format
/
BOOL fActivate;
/* server activation flag
/
```

The OleCreateInvisible function creates an object without displaying the server application to the user. The function either starts the server to create the object or creates a blank object of the specified class and format without starting the server.

Parameter	Description								
lpszProtocol	Points to a null-terminated string specifying the name of the protocol required for the new embedded object. Currently, this value can be StdFileEditing (the name of the object linking and embedding protocol) or Static (for uneditable pictures only).								
lpClient	Points to an OLECLIENT structure allocated and initialized by the client application. This pointer is used to locate the callback function and is passed in callback notifications.								
lpszClass	Points to a null-terminated string specifying the registered name of the class of the object to be created.								
lhClientDoc	Identifies the client document in which the object is being created.								
lpszObjname	Points to a null-terminated string specifying the client's name for the object. This name must be unique with respect to the names of any other objects in the document and cannot contain a slash mark (/).								
lplpObject	Points to a variable where the library will store the long pointer to the new object.								
renderopt	Specifies the client's preference for presentation data for the object. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>olerender_draw</td><td>The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.</td></tr><tr><td>olerender_format</td><td>The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.</td></tr><tr><td>olerender_none</td><td>The client library does not obtain any presentation data and does not draw the object.</td></tr></tbody></table>	Value	Meaning	olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.	olerender_format	The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.	olerender_none	The client library does not obtain any presentation data and does not draw the object.
Value	Meaning								
olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.								
olerender_format	The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.								
olerender_none	The client library does not obtain any presentation data and does not draw the object.								

**cfFormat** Specifies the clipboard format when the `renderopt` parameter is `olerender_format`. This clipboard format is used in a subsequent call to `OleGetData`. If this clipboard format is `CF_METAFILEPICT`, `CF_DIB`, or `CF_BITMAP`, the library manages the data and draws the object. The library does not support drawing for any other formats.

**fActivate** Specifies whether to start the server for the object. If this parameter is `TRUE` the server is started (but not shown). If this parameter is `FALSE`, the server is not started and the function creates a blank object of the specified class and format.

#### Returns

The return value is `OLE_OK` if the function is successful. Otherwise, it is an error value, which may be one of the following:

`OLE_ERROR_HANDLE`  
`OLE_ERROR_NAME`  
`OLE_ERROR_PROTOCOL`

#### Comments

An application can avoid redrawing an object repeatedly by calling the `OleCreateInvisible` function before using such functions as `OleSetBounds`, `OleSetColorScheme`, and `OleSetTargetDevice` to set up the object. After setting up the object, the application can either call the `OleActivate` function to display the object or call the `OleUpdate` and `OleClose` functions to update the object without displaying it.

#### See Also

`OleActivate`, `OleClose`, `OleSetBounds`, `OleSetColorScheme`, `OleSetTargetDevice`, `OleUpdate`

### OleCreateLinkFromClip (3.1)

```
#include ole.h

OLESTATUS OleCreateLinkFromClip(lpszProtocol, lpClient, lhClientDoc, lpszObjname, lpIpObject,
                                renderopt, cfFormat)
LPCSTR lpszProtocol;          /* address of string for protocol name */
LPOLECLIENT lpClient;        /* address of client structure */
LHCLIENTDOC lhClientDoc;     /* long handle of client document */
LPCSTR lpszObjname;          /* address of string for object name */
LPOLEOBJECT FAR* lpIpObject; /* address of pointer to object */
OLEOPT_RENDER renderopt;     /* rendering options */
OLECLIPFORMAT cfFormat;     /* clipboard format
```

The OleCreateLinkFromClip function typically creates a link to an object from the clipboard.

Parameter	Description								
lpszProtocol	Points to a null-terminated string specifying the name of the required protocol. Currently, this value can be StdFileEditing (the name of the object linking and embedding protocol).								
lpClient	Points to an <u>OLECLIENT</u> structure allocated and initialized by the client application. This pointer is used to locate the callback function and is passed in callback notifications.								
lhClientDoc	Identifies the client document in which the object is being created.								
lpszObjname	Points to a null-terminated string specifying the client's name for the object. This name must be unique with respect to the names of any other objects in the document and cannot contain a slash mark (/).								
lpIpObject	Points to a variable where the library will store the long pointer to the new object.								
renderopt	Specifies the client's preference for presentation data for the object. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>olerender_draw</td><td>The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.</td></tr><tr><td>olerender_format</td><td>The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.</td></tr><tr><td>olerender_none</td><td>The client library does not obtain any presentation data and does not draw the object.</td></tr></tbody></table>	Value	Meaning	olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.	olerender_format	The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.	olerender_none	The client library does not obtain any presentation data and does not draw the object.
Value	Meaning								
olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.								
olerender_format	The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.								
olerender_none	The client library does not obtain any presentation data and does not draw the object.								
cfFormat	Specifies the clipboard format when the renderopt parameter is olerender_format. This clipboard format is used in a subsequent call to <u>OleGetData</u> . If this clipboard format is CF_METAFILEPICT, CF_DIB, or CF_BITMAP, the library manages the data and draws the object. The library does not support drawing for any other formats.								

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_CLIP  
OLE\_ERROR\_FORMAT

OLE\_ERROR\_HANDLE  
OLE\_ERROR\_NAME  
OLE\_ERROR\_PROTOCOL  
OLE\_WAIT\_FOR\_RELEASE

#### Comments

The `olerender_none` rendering option is typically used to support hyperlinks. With this option, the client does not call the OleDraw function and calls OleGetData only for ObjectLink, OwnerLink, and Native formats.

The `olerender_format` rendering option allows a client to compute data (instead of painting it), use an unusual data format, or modify a standard data format. With this option, the client does not call OleDraw. The client calls OleGetData to retrieve data in the specified format.

The `olerender_draw` rendering option is the most typical option. It is the easiest rendering option for the client to implement (the client simply calls OleDraw), and it allows the most flexibility. An object handler can exploit this flexibility to store no presentation data, a private presentation data format, or several different formats that it can choose among dynamically. Future implementations of object linking and embedding (OLE) may also exploit the flexibility that is inherent in this option.

#### See Also

OleCreate, OleCreateFromTemplate, OleDraw, OleGetData, OleQueryLinkFromClip

### OleCreateLinkFromFile (3.1)

```
#include ole.h

OLESTATUS OleCreateLinkFromFile(lpszProtocol, lpClient, lpszClass, lpszFile, lpszItem, lhClientDoc,
                                lpszObjname, lp lpObject, renderopt, cfFormat)
LPCSTR lpszProtocol;          /* address of string for protocol name */
LPOLECLIENT lpClient;
* address of client structure
/
LPCSTR lpszClass;
* string for class name
/
LPCSTR lpszFile;
* address of string for filename
/
LPCSTR lpszItem;
* address of string for doc. part to link
/
LHCLIENTDOC lhClientDoc;
* long handle of client document
/
LPCSTR lpszObjname;
* address of string for object name
/
LPOLEOBJECT FAR* lp lpObject;
* address of pointer to new object
/
OLEOPT_RENDER renderopt;
* rendering options
/
OLECLIPFORMAT cfFormat;
* clipboard format
/
```

The OleCreateLinkFromFile function creates a linked object from a file that contains an object. If necessary, the library starts the server to render the presentation data, but the object is not shown in the server for editing.

Parameter	Description				
lpszProtocol	Points to a null-terminated string specifying the name of the required protocol. Currently, this value can be StdFileEditing (the name of the object linking and embedding protocol).				
lpClient	Points to an OLECLIENT structure allocated and initialized by the client application. This pointer is used to locate the callback function and is passed in callback notifications.				
lpszClass	Points to a null-terminated string specifying the name of the class for the new object. If this value is NULL, the library uses the extension of the filename pointed to by the lpszFile parameter to find the class name for the object.				
lpszFile	Points to a null-terminated string specifying the name of the file containing the object.				
lpszItem	Points to a null-terminated string identifying the part of the document to link to. If this value is NULL, the link is to the entire document.				
lhClientDoc	Identifies the client document in which the object is being created.				
lpszObjname	Points to a null-terminated string specifying the client's name for the object. This name must be unique with respect to the names of any other objects in the document and cannot contain a slash mark (/).				
lp lpObject	Points to a variable where the library will store the long pointer to the new object.				
renderopt	Specifies the client's preference for presentation data for the object. This parameter can be one of the following values:				
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>olerender_draw</td><td>The client calls the OleDraw function, and the library obtains and manages presentation data.</td></tr></tbody></table>	Value	Meaning	olerender_draw	The client calls the OleDraw function, and the library obtains and manages presentation data.
Value	Meaning				
olerender_draw	The client calls the OleDraw function, and the library obtains and manages presentation data.				



olerender_format	The client calls the <u>OleGetData</u> function to retrieve data in a specific format. The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.
olerender_none	The client library does not obtain any presentation data and does not draw the object.
cfFormat	Specifies the clipboard format when the renderopt parameter is olerender_format. This clipboard format is used in a subsequent call to <u>OleGetData</u> . If this clipboard format is <u>CF_METAFILEPICT</u> , <u>CF_DIB</u> , or <u>CF_BITMAP</u> , the library manages the data and draws the object. The library does not support drawing for any other formats.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_CLASS  
OLE\_ERROR\_HANDLE  
OLE\_ERROR\_MEMORY  
OLE\_ERROR\_NAME  
OLE\_ERROR\_PROTOCOL  
OLE\_WAIT\_FOR\_RELEASE

#### Comments

The olerender\_none rendering option is typically used to support hyperlinks. With this option, the client does not call OleDraw and calls OleGetData only for ObjectLink, OwnerLink, and Native formats.

The olerender\_format rendering option allows a client to compute data (instead of painting it), use an unusual data format, or modify a standard data format. With this option, the client does not call OleDraw. The client calls OleGetData to retrieve data in the specified format.

The olerender\_draw rendering option is the most typical option. It is the easiest rendering option for the client to implement (the client simply calls OleDraw), and it allows the most flexibility. An object handler can exploit this flexibility to store no presentation data, a private presentation data format, or several different formats that it can choose among dynamically. Future implementations of object linking and embedding (OLE) may also exploit the flexibility that is inherent in this option.

#### See Also

OleCreate, OleCreateFromFile, OleCreateFromTemplate, OleDraw, OleGetData

## OleDelete (3.1)

#include ole.h

```
OLESTATUS OleDelete(lpObject)
LPOLEOBJECT lpObject;      /* address of object to delete */
```

The OleDelete function deletes an object and frees memory that was associated with that object. If the object was open, it is closed.

Parameter	Description
lpObject	Points to the object to delete.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_BUSY
OLE_ERROR_OBJECT
OLE_WAIT_FOR_RELEASE
```

### Comments

An application uses the OleDelete function when the object is no longer part of the client document.

The OleDelete function, unlike OleRelease, indicates that the object has been permanently removed.

### See Also

OleClose, OleRelease

## OleDraw (3.1)

```
#include ole.h

OLESTATUS OleDraw(lpObject, hdc, lprcBounds, lprcWBounds, hdcFormat)
LPOLEOBJECT lpObject;          /* address of object to draw          */
HDC hdc;                        /* handle of DC for drawing object    */
/* handle of DC for drawing object
/
const RECT FAR* lprcBounds;
/* bounding rectangle for drawing object
/
const RECT FAR* lprcWBounds;
/* bounding rectangle for metafile DC
/
HDC hdcFormat;
/* handle of DC for formatting object
/
```

The OleDraw function draws a specified object into a bounding rectangle in a device context.

Parameter	Description
lpObject	Points to the object to draw.
hdc	Identifies the device context in which to draw the object.
lprcBounds	Points to a <u>RECT</u> structure defining the bounding rectangle, in logical units for the device context specified by the hdc parameter, in which to draw the object.
lprcWBounds	Points to a <u>RECT</u> structure defining the bounding rectangle if the hdc parameter specifies a <u>metafile</u> . The left and top members of the RECT structure should specify the window origin, and the right and bottom members should specify the window extents.
hdcFormat	Identifies a device context describing the target device for which to format the object.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_ERROR_ABORT
OLE_ERROR_BLANK
OLE_ERROR_DRAW
OLE_ERROR_MEMORY
OLE_ERROR_OBJECT
```

### Comments

This function returns OLE\_ERROR\_ABORT if the callback function returns FALSE during drawing.

When the hdc parameter specifies a metafile device context, the rectangle specified by the lprcWBounds parameter contains the rectangle specified by the lprcBounds parameter. If hdc does not specify a metafile device context, the lprcWBounds parameter is ignored.

The library may use an object handler to render the object, and this object handler may need information about the target device. Therefore, the device-context handle specified by the hdcFormat parameter is required. The lprcBounds parameter identifies the rectangle on the device context (relative to its current mapping mode) that the object should be mapped onto. This may involve scaling the picture and can be used by client applications to impose a view scaling between the displayed view and the final printed image.

An object handler should format an object as if it were to be drawn at the size specified by a call to the OleSetBounds function for the device context specified by the hdcFormat parameter. Often this formatting will already have been done by the server application; in this case, the library simply renders the presentation data with suitable scaling for the required bounding rectangle. If cropping or banding is required, the device context in which the object is drawn may include a clipping region smaller than the specified bounding rectangle.

### See Also

[OleSetBounds](#)

## OleEnumFormats (3.1)

#include ole.h

```
OLECLIPFORMAT OleEnumFormats(lpObject, cfFormat)
LPOLEOBJECT lpObject;          /* address of object to query */
OLECLIPFORMAT cfFormat;
/* format from previous function call
/
```

The OleEnumFormats function enumerates the data formats that describe a specified object.

Parameter	Description
lpObject	Points to the object to be queried.
cfFormat	Specifies the format returned by the last call to the OleEnumFormats function. For the first call to this function, this parameter is zero.

### Returns

The return value is the next available format if any further formats are available. Otherwise, the return value is NULL.

### Comments

When an application specifies NULL for the cfFormat parameter, the OleEnumFormats function returns the first available format. Whenever an application specifies a format that was returned by a previous call to OleEnumFormats, the function returns the next available format, in sequence. When no more formats are available, the function returns NULL.

### See Also

[OleGetData](#)

### OleEnumObjects (3.1)

#include ole.h

```
OLESTATUS OleEnumObjects(lhDoc, lplpObject)
LHCLIENTDOC lhDoc; /* document handle */
LPOLEOBJECT FAR* lplpObject;
/* address of pointer to object
/
```

The OleEnumObjects function enumerates the objects in a specified document.

Parameter	Description
lhDoc	Identifies the document for which the objects are enumerated.
lplpObject	Points to an object in the document when the function returns. For the first call to this function, this parameter should point to a NULL object.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_HANDLE  
OLE\_ERROR\_OBJECT

#### Comments

When an application specifies a NULL object for the lplpObject parameter, the OleEnumObjects function returns the first object in the document. Whenever an application specifies an object that was returned by a previous call to OleEnumObjects, the function returns the next object, in sequence. When there are no more objects in the document, the lplpObject parameter points to a NULL object.

Only objects that have been loaded and not released are enumerated by this function.

#### See Also

OleDelete, OleRelease

### OleEqual (3.1)

#include ole.h

```
OLESTATUS OleEqual(lpObject1, lpObject2)
```

```
LPOLEOBJECT lpObject1; /* address of first object to compare */
```

```
LPOLEOBJECT lpObject2;
```

```
/* address of second object to compare
```

```
/
```

The OleEqual function compares two objects for equality.

Parameter	Description
lpObject1	Points to the first object to test for equality.
lpObject2	Points to the second object to test for equality.

#### Returns

The return value is OLE\_OK if the specified objects are equal. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_OBJECT

OLE\_ERROR\_NOT\_EQUAL

#### Comments

Embedded objects are equal if their class, item, and native data are identical. Linked objects are equal if their class, document, and item are identical.

#### See Also

[OleClone](#), [OleQueryOutOfDate](#)

## OleExecute (3.1)

#include ole.h

```
OLESTATUS OleExecute(lpObject, hglbCmds, reserved)
LPOLEOBJECT lpObject; /* address of object receiving DDE commands */
HGLOBAL hglbCmds;
/* handle of memory with commands
/
UINT reserved;
* reserved
/
```

The OleExecute function sends dynamic data exchange (DDE) execute commands to the server for the specified object.

Parameter	Description
lpObject	Points to an object identifying the server to which DDE execute commands are sent.
hglbCmds	Identifies the memory containing one or more DDE execute commands.
reserved	Reserved; must be zero.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_BUSY
OLE_ERROR_COMMAND
OLE_ERROR_MEMORY
OLE_ERROR_NOT_OPEN
OLE_ERROR_OBJECT
OLE_ERROR_PROTOCOL
OLE_ERROR_STATIC
OLE_WAIT_FOR_RELEASE
```

### Comments

The client application should call the OleQueryProtocol function, specifying StdExecute, before calling the OleExecute function. The OleQueryProtocol function succeeds if the server for an object supports the OleExecute function.

### See Also

OleQueryProtocol

### OleGetData (3.1)

```
#include ole.h

OLESTATUS OleGetData(lpObject, cfFormat, lphData)
LPOLEOBJECT lpObject;          /* address of object to query */
OLECLIPFORMAT cfFormat;
/* format for retrieved data */
/
HANDLE FAR* lphData;
/* address of memory to contain data */
/
```

The OleGetData function retrieves data in the requested format from the specified object and supplies the handle of a memory or graphics device interface (GDI) object containing the data.

Parameter	Description
lpObject	Points to the object from which data is retrieved.
cfFormat	Specifies the format in which data is returned. This parameter can be one of the predefined clipboard formats or the value returned by the <a href="#">RegisterClipboardFormat</a> function.
lphData	Points to the handle of a memory object that contains the data when the function returns.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_ERROR_BLANK
OLE_ERROR_FORMAT
OLE_ERROR_OBJECT
OLE_WARN_DELETE_DATA
```

#### Comments

If the OleGetData function returns OLE\_WARN\_DELETE\_DATA, the client application owns the data and should free the memory associated with the data when the client has finished using it. For other return values, the client should not free the memory or modify the data, because the data is controlled by the client library. If the application needs the data for long-term use, it should copy the data.

The OleGetData function typically returns OLE\_WARN\_DELETE\_DATA if an object handler generates data for an object that the client library cannot interpret. In this case, the client application is responsible for controlling that data.

When the OleGetData function specifies CF\_METAFILE or CF\_BITMAP, the lphData parameter points to a GDI object, not a memory object, when the function returns. OleGetData supplies the handle of a memory object for all other formats.

#### See Also

[OleEnumFormats](#), [OleSetData](#), [RegisterClipboardFormat](#)



## OleGetLinkUpdateOptions (3.1)

#include ole.h

```
OLESTATUS OleGetLinkUpdateOptions(lpObject, lpUpdateOpt)
LPOLEOBJECT lpObject;          /* address of object to query */
OLEOPT_UPDATE FAR* lpUpdateOpt;
/* address of update options
/
```

The OleGetLinkUpdateOptions function retrieves the link-update options for the presentation of a specified object.

Parameter	Description								
lpObject	Points to the object to query.								
lpUpdateOpt	Points to a variable in which the function stores the current value of the link-update option for the specified object. The link-update option setting may be one of the following values:								
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>oleupdate_always</td><td>Update the linked object whenever possible. This option supports the Automatic link-update radio button in the Links dialog box.</td></tr><tr><td>oleupdate_onsave</td><td>Update the linked object only on request from the client application. This option supports the Manual link-update radio button in the Links dialog box.</td></tr><tr><td>oleupdate_onsave</td><td>Update the linked object when the source document is saved by the server.</td></tr></tbody></table>	Value	Meaning	oleupdate_always	Update the linked object whenever possible. This option supports the Automatic link-update radio button in the Links dialog box.	oleupdate_onsave	Update the linked object only on request from the client application. This option supports the Manual link-update radio button in the Links dialog box.	oleupdate_onsave	Update the linked object when the source document is saved by the server.
Value	Meaning								
oleupdate_always	Update the linked object whenever possible. This option supports the Automatic link-update radio button in the Links dialog box.								
oleupdate_onsave	Update the linked object only on request from the client application. This option supports the Manual link-update radio button in the Links dialog box.								
oleupdate_onsave	Update the linked object when the source document is saved by the server.								

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_OBJECT  
OLE\_ERROR\_STATIC

### See Also

[OleSetLinkUpdateOptions](#)

### OleIsDcMeta (3.1)

```
#include ole.h
```

```
BOOL OleIsDcMeta(hdc)
```

```
HDC hdc; /* device-context handle */
```

The OleIsDcMeta function determines whether the specified device context is a metafile device context.

Parameter	Description
hdc	Identifies the device context to query.

#### Returns

The return value is a positive value if the device context is a metafile device context. Otherwise, it is NULL.

### OleLoadFromStream (3.1)

```
#include ole.h
```

```
OLESTATUS OleLoadFromStream(lpStream, lpszProtocol, lpClient, lhClientDoc, lpszObjname,  
                             lpObject)
```

```
LPOLESTREAM lpStream;          /* address of stream for object */
```

```
LPCSTR lpszProtocol;
```

```
/* address of string for protocol name
```

```
/
```

```
LPOLECLIENT lpClient;
```

```
/* address of client structure
```

```
/
```

```
LHCLIENTDOC lhClientDoc;
```

```
/* long handle of client document
```

```
/
```

```
LPCSTR lpszObjname;
```

```
/* address of string for object name
```

```
/
```

```
LPOLEOBJECT FAR* lpObject;
```

```
/* address of pointer to object
```

```
/
```

The OleLoadFromStream function loads an object from the containing document.

Parameter	Description
lpStream	Points to an <u>OLESTREAM</u> structure that was allocated and initialized by the client application. The library calls the Get function in the <u>OLESTREAMVTBL</u> structure to obtain the data for the object.
lpszProtocol	Points to a null-terminated string specifying the name of the required protocol. Currently, this value can be StdFileEditing (the name of the object linking and embedding protocol) or Static (for uneditable pictures only).
lpClient	Points to an <u>OLECLIENT</u> structure allocated and initialized by the client application. This pointer is used to locate the callback function and is passed in callback notifications.
lhClientDoc	Identifies the client document in which the object is being created.
lpszObjname	Points to a null-terminated string specifying the client's name for the object.
lpObject	Points to a variable in which the library stores a pointer to the loaded object.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_ERROR_HANDLE  
OLE_ERROR_NAME  
OLE_ERROR_PROTOCOL  
OLE_ERROR_STREAM  
OLE_WAIT_FOR_RELEASE
```

#### Comments

To load an object, the client application needs only the location of that object in a file. A client typically loads an object only when the object is needed (for example, when it must be displayed).

If an object cannot be loaded when the lpszProtocol parameter specifies StdFileEditing, the application can call the OleLoadFromStream function again, specifying Static.

If the object is linked and the server and document are open, the library automatically makes the link between the client and server applications when an application calls OleLoadFromStream.

#### See Also

OleQuerySize, OleSaveToStream

## OleLockServer (3.1)

```
#include ole.h
```

```
OLESTATUS OleLockServer(lpObject, lphServer)
LPOLEOBJECT lpObject;          /* address of object */
LHSERVER FAR* lphServer;      /* address of handle of server */
/
```

The OleLockServer function is called by a client application to keep an open server application in memory. Keeping the server application in memory allows the client library to use the server application to open objects quickly.

Parameter	Description
lpObject	Points to an object the client library uses to identify the open server application to keep in memory. When the server has been locked, this object can be deleted.
lphServer	Points to the handle of the server application when the function returns.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_ERROR_COMM
OLE_ERROR_LAUNCH
OLE_ERROR_OBJECT
```

### Comments

A client calls OleLockServer to speed the opening of objects when the same server is used for a number of different objects. Before the client terminates, it must call the OleUnlockServer function to release the server from memory.

When OleLockServer is called more than once for a given server, even by different client applications, the server's lock count is increased. Each call to OleUnlockServer decrements the lock count. The server remains locked until the lock count is zero. If the object identified by the lpObject parameter is deleted before calling the OleUnlockServer function, OleUnlockServer must still be called to decrement the lock count.

If necessary, a server can terminate even though a client has called the OleLockServer function.

### See Also

OleUnlockServer

### OleObjectConvert (3.1)

#include ole.h

```
OLESTATUS OleObjectConvert(lpObject, lpszProtocol, lpClient, lhClientDoc, lpszObjname,  
                           lpIpObject)
```

```
LPOLEOBJECT lpObject;          /* address of object to convert      */
```

```
LPCSTR lpszProtocol;
```

```
/* address of string for protocol name
```

```
/
```

```
LPOLECLIENT lpClient;
```

```
/* address of client for new object
```

```
/
```

```
LHCLIENTDOC lhClientDoc;
```

```
/* long handle of client document
```

```
/
```

```
LPCSTR lpszObjname;
```

```
/* address of string for object name
```

```
/
```

```
LPOLEOBJECT FAR* lpIpObject;
```

```
/* address of pointer to new object
```

```
/
```

The OleObjectConvert function creates a new object that supports a specified protocol by converting an existing object. This function neither deletes nor replaces the original object.

Parameter	Description
lpObject	Points to the object to convert.
lpszProtocol	Points to a null-terminated string specifying the name of the required protocol. Currently this value can be Static (for uneditable pictures only).
lpClient	Points to an <u>OLECLIENT</u> structure for the new object.
lhClientDoc	Identifies the client document in which the object is being created.
lpszObjname	Points to a null-terminated string specifying the client's name for the object. This name must be unique with respect to the names of any other objects in the document and cannot contain a slash mark (/).
lpIpObject	Points to a variable in which the library stores a pointer to the new object.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_BUSY  
OLE_ERROR_HANDLE  
OLE_ERROR_NAME  
OLE_ERROR_OBJECT  
OLE_ERROR_STATIC
```

#### Comments

The only conversion currently supported is that of changing a linked or embedded object to a static object.

See Also

OleClone

### OleQueryBounds (3.1)

#include ole.h

```
OLESTATUS OleQueryBounds(lpObject, lpBounds)
LPOLEOBJECT lpObject; /* address of object to query */
RECT FAR* lpBounds;
/* address of structure for bounding rectangle */
```

The OleQueryBounds function retrieves the extents of the bounding rectangle on the target device for the specified object. The coordinates are in MM\_HIMETRIC units.

Parameter	Description										
lpObject	Points to the object to query.										
lpBounds	Points to a <u>RECT</u> structure for the extents of the bounding rectangle. The members of the <u>RECT</u> structure have the following meanings:										
	<table><thead><tr><th>Member</th><th>Meaning</th></tr></thead><tbody><tr><td>rect.left</td><td>0</td></tr><tr><td>rect.top</td><td>0</td></tr><tr><td>rect.right</td><td>x-extent</td></tr><tr><td>rect.bottom</td><td>y-extent</td></tr></tbody></table>	Member	Meaning	rect.left	0	rect.top	0	rect.right	x-extent	rect.bottom	y-extent
Member	Meaning										
rect.left	0										
rect.top	0										
rect.right	x-extent										
rect.bottom	y-extent										

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_BLANK  
OLE\_ERROR\_MEMORY  
OLE\_ERROR\_OBJECT

#### See Also

OleSetBounds, SetMapMode, RECT

## OleQueryClientVersion (3.1)

#include ole.h

DWORD OleQueryClientVersion(void)

The OleQueryClientVersion function retrieves the version number of the client library.

### Returns

The return value is a doubleword value. The major version number is in the low-order byte of the low-order word, and the minor version number is in the high-order byte of the low-order word. The high-order word is reserved.

### See Also

[OleQueryServerVersion](#)

### OleQueryCreateFromClip (3.1)

```
#include ole.h

OLESTATUS OleQueryCreateFromClip(lpszProtocol, renderopt, cfFormat)
LPCSTR lpszProtocol; /* address of string for protocol name */
OLEOPT_RENDER renderopt;
/* rendering options
/
OLECLIPFORMAT cfFormat;
/* format for clipboard data
/
```

The OleQueryCreateFromClip function checks whether the object on the clipboard supports the specified protocol and rendering options.

Parameter	Description								
lpszProtocol	Points to a null-terminated string specifying the name of the protocol needed by the client. Currently, this value can be StdFileEditing (the name of the object linking and embedding protocol) or Static (for uneditable pictures only).								
renderopt	Specifies the client's preference for presentation data for the object. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>olerender_draw</td><td>The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.</td></tr><tr><td>olerender_format</td><td>The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.</td></tr><tr><td>olerender_none</td><td>The client library does not obtain any presentation data and does not draw the object.</td></tr></tbody></table>	Value	Meaning	olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.	olerender_format	The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.	olerender_none	The client library does not obtain any presentation data and does not draw the object.
Value	Meaning								
olerender_draw	The client calls the <u>OleDraw</u> function, and the library obtains and manages presentation data.								
olerender_format	The library obtains and manages the data in the requested format, as specified by the cfFormat parameter.								
olerender_none	The client library does not obtain any presentation data and does not draw the object.								
cfFormat	Specifies the clipboard format. This parameter is used only when the renderopt parameter is olerender_format. If the clipboard format is CF_METAFILEPICT, CF_DIB, or CF_BITMAP, the library manages the data and draws the object. The library does not support drawing for any other formats.								

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_FORMAT  
OLE\_ERROR\_PROTOCOL

#### Comments

The OleQueryCreateFromClip function is typically used to check whether to enable a Paste command.

The olerender\_none rendering option is typically used to support hyperlinks. With this option, the client does not call OleDraw and calls the OleGetData function only for ObjectLink, OwnerLink, and Native formats.

The olerender\_format rendering option allows a client to compute data (instead of painting it), use an unusual data format, or modify a standard data format. With this option the client does not call OleDraw. The client calls OleGetData to retrieve data in the specified format.

The olerender\_draw rendering option is the most typical option. It is the easiest rendering option for the client to implement (the client simply calls OleDraw), and it allows the most flexibility. An object handler can exploit this flexibility to store no presentation data, a private presentation data format, or several different formats that it can choose among dynamically. Future implementations of object linking and embedding (OLE) may also exploit the flexibility that is inherent in this option.

#### See Also

OleCreateFromClip, OleDraw, OleGetData



### OleQueryLinkFromClip (3.1)

```
#include ole.h

OLESTATUS OleQueryLinkFromClip(lpszProtocol, renderopt, cfFormat)
LPCSTR lpszProtocol; /* address of string for protocol name */
OLEOPT_RENDER renderopt;
/* rendering options
/
OLECLIPFORMAT cfFormat;
/* format for clipboard data
/
```

The `OleQueryLinkFromClip` function checks whether a client application can use the data on the clipboard to produce a linked object that supports the specified protocol and rendering options.

Parameter	Description								
<code>lpszProtocol</code>	Points to a null-terminated string specifying the name of the protocol needed by the client. Currently this value can be <code>StdFileEditing</code> (the name of the object linking and embedding protocol).								
<code>renderopt</code>	Specifies the client's preference for presentation data for the object. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><code>olerender_draw</code></td><td>The client calls the <code>OleDraw</code> function, and the library obtains and manages presentation data.</td></tr><tr><td><code>olerender_format</code></td><td>The library obtains and manages the data in the requested format, as specified by the <code>cfFormat</code> parameter.</td></tr><tr><td><code>olerender_none</code></td><td>The client library does not obtain any presentation data and does not draw the object.</td></tr></tbody></table>	Value	Meaning	<code>olerender_draw</code>	The client calls the <code>OleDraw</code> function, and the library obtains and manages presentation data.	<code>olerender_format</code>	The library obtains and manages the data in the requested format, as specified by the <code>cfFormat</code> parameter.	<code>olerender_none</code>	The client library does not obtain any presentation data and does not draw the object.
Value	Meaning								
<code>olerender_draw</code>	The client calls the <code>OleDraw</code> function, and the library obtains and manages presentation data.								
<code>olerender_format</code>	The library obtains and manages the data in the requested format, as specified by the <code>cfFormat</code> parameter.								
<code>olerender_none</code>	The client library does not obtain any presentation data and does not draw the object.								
<code>cfFormat</code>	Specifies the clipboard format. This parameter is used only when the <code>renderopt</code> parameter is <code>olerender_format</code> . If this clipboard format is <code>CF_METAFILEPICT</code> , <code>CF_DIB</code> , or <code>CF_BITMAP</code> , the library manages the data and draws the object. The library does not support drawing for any other formats.								

#### Returns

The return value is `OLE_OK` if the function is successful. Otherwise, it is an error value, which may be one of the following:

`OLE_ERROR_FORMAT`  
`OLE_ERROR_PROTOCOL`

#### Comments

The `OleQueryLinkFromClip` function is typically used to check whether to enable a Paste Link command.

The `olerender_none` rendering option is typically used to support hyperlinks. With this option, the client does not call `OleDraw` and calls the `OleGetData` function only for ObjectLink, OwnerLink, and Native formats.

The `olerender_format` rendering option allows a client to compute data (instead of painting it), use an unusual data format, or modify a standard data format. With this option, the client does not call `OleDraw`. The client calls `OleGetData` to retrieve data in the specified format.

The `olerender_draw` rendering option is the most typical option. It is the easiest rendering option for the client to implement (the client simply calls `OleDraw`), and it allows the most flexibility. An object handler can exploit this flexibility to store no presentation data, a private presentation data format, or several different formats that it can choose among dynamically. Future implementations of object linking and embedding (OLE) may also exploit the flexibility that is inherent in this option.

#### See Also

[OleCreateLinkFromClip](#), [OleDraw](#), [OleGetData](#)

### OleQueryName (3.1)

#include ole.h

```
OLESTATUS OleQueryName(lpObject, lpszObject, lpwBuffSize)
LPOLEOBJECT lpObject;          /* address of object          */
LPSTR lpszObject;              /* address of string for object name */
/* address of word for size of buffer */
```

The OleQueryName function retrieves the name of a specified object.

Parameter	Description
lpObject	Points to the object whose name is being queried.
lpszObject	Points to a character array that contains a null-terminated string. When the function returns, this string specifies the name of the object.
lpwBuffSize	Points to a variable containing the size, in bytes, of the buffer pointed to by the lpszObject parameter. When the function returns, this value is the number of bytes copied to the buffer.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be OLE\_ERROR\_OBJECT.

#### See Also

OleRename

## OleQueryOpen (3.1)

#include ole.h

```
OLESTATUS OleQueryOpen(lpObject)
LPOLEOBJECT lpObject;          /* address of object to query */
```

The OleQueryOpen function checks whether the specified object is open.

Parameter	Description
lpObject	Points to the object to query.

### Returns

The return value is OLE\_OK if the object is open. Otherwise, it is an error value, which may be one of the following:

```
OLE_ERROR_COMM
OLE_ERROR_OBJECT
OLE_ERROR_STATIC
```

See Also

[OleActivate](#)

## OleQueryOutOfDate (3.1)

#include ole.h

```
OLESTATUS OleQueryOutOfDate(lpObject)
LPOLEOBJECT lpObject; /* address of object to query */
```

The OleQueryOutOfDate function checks whether an object is out-of-date.

Parameter	Description
lpObject	Points to the object to query.

### Returns

The return value is OLE\_OK if the object is up-to-date. Otherwise, it is an error value, which may be one of the following:

```
OLE_ERROR_OBJECT
OLE_ERROR_OUTOFDATE
```

### Comments

The OleQueryOutOfDate function has not been implemented for the current version of object linking and embedding (OLE). For linked objects, OleQueryOutOfDate always returns OLE\_OK.

A linked object might be out-of-date if the document that is the source for the link has been updated. An embedded object that contains links to other objects might also be out-of-date.

### See Also

OleEqual, OleUpdate

## OleQueryProtocol (3.1)

#include ole.h

```
void FAR* OleQueryProtocol(lpobj, lpszProtocol)
LPOLEOBJECT lpobj;          /* address of object to query          */
LPCSTR lpszProtocol;        /* address of string for protocol to query */
/
```

The OleQueryProtocol function checks whether an object supports a specified protocol.

Parameter	Description
lpobj	Points to the object to query.
lpszProtocol	Points to a null-terminated string specifying the name of the requested protocol. This value can be StdFileEditing or StdExecute.

### Returns

The return value is a void pointer to an OLEOBJECT structure if the function is successful, or it is NULL if the object does not support the requested protocol. The library can return OLE\_WAIT\_FOR\_RELEASE when an application calls this function.

### Comments

The OleQueryProtocol function queries whether the specified protocol is supported and returns a modified object pointer that allows access to the function table for the protocol. This modified object pointer points to a structure that has the same form as the OLEOBJECT structure; the new structure also points to a table of functions and may contain additional state information. The new pointer does not point to a different object--if the object is deleted, secondary pointers become invalid. If a protocol includes delete functions, calling a delete function invalidates all pointers to that object.

A client application typically calls OleQueryProtocol, specifying StdExecute for the lpszProtocol parameter, before calling the OleExecute function. This allows the client application to check whether the server for an object supports dynamic data exchange (DDE) execute commands.

### See Also

OleExecute

## OleQueryReleaseError (3.1)

#include ole.h

```
OLESTATUS OleQueryReleaseError(lpobj)
LPOLEOBJECT lpobj;          /* address of object to query */
```

The OleQueryReleaseError function checks the error value for an asynchronous operation on an object.

Parameter	Description
lpobj	Points to an object for which the error value is to be queried.

### Returns

The return value, if the function is successful, is either OLE\_OK if the asynchronous operation completed successfully or the error value for that operation. If the pointer passed in the lpobj parameter is invalid, the function returns OLE\_ERROR\_OBJECT.

### Comments

A client application receives the OLE\_RELEASE notification when an asynchronous operation has terminated. The client should then call OleQueryReleaseError to check whether the operation has terminated successfully or with an error value.

### See Also

[OleQueryReleaseMethod](#), [OleQueryReleaseStatus](#)

## OleQueryReleaseMethod (3.1)

#include ole.h

```
OLE_RELEASE_METHOD OleQueryReleaseMethod(lpobj)  
LPOLEOBJECT lpobj; /* address of object to query */
```

The OleQueryReleaseMethod function finds out the operation that finished for the specified object.

Parameter	Description
lpobj	Points to an object for which the operation is to be queried.

Returns

The return value indicates the server operation (method) that finished. It can be one of the following values:

Value	Server operation
OLE_ACTIVATE	Activate
OLE_CLOSE	Close
OLE_COPYFROMLNK	CopyFromLink (autoreconnect)
OLE_CREATE	Create
OLE_CREATEFROMFILE	CreateFromFile
OLE_CREATEFROMTEMPLATE	CreateFromTemplate
OLE_CREATEINVISIBLE	CreateInvisible
OLE_CREATELINKFROMFILE	CreateLinkFromFile
OLE_DELETE	Object Delete
OLE_EMBPASTE	Paste and Update
OLE_LNKPASTE	PasteLink (autoreconnect)
OLE_LOADFROMSTREAM	LoadFromStream (autoreconnect)
OLE_NONE	No operation active
OLE_OTHER	Other miscellaneous asynchronous operations
OLE_RECONNECT	Reconnect
OLE_REQUESTDATA	OleRequestData
OLE_RUN	Run
OLE_SERVERUNLAUNCH	Server is stopping
OLE_SETDATA	OleSetData
OLE_SETUPDATEOPTIONS	Setting update options
OLE_SHOW	Show
OLE_UPDATE	Update

If the pointer passed in the lpobj parameter is invalid, the function returns OLE\_ERROR\_OBJECT.

Comments

A client application receives the OLE\_RELEASE notification when an asynchronous operation has ended. The client can then call OleQueryReleaseMethod to check which operation caused the library to send the OLE\_RELEASE notification. The client calls OleQueryReleaseError to determine whether the operation terminated successfully or with an error value.

See Also

OleQueryReleaseError, OleQueryReleaseStatus

## OleQueryReleaseStatus (3.1)

#include ole.h

```
OLESTATUS OleQueryReleaseStatus(lpobj)  
LPOLEOBJECT lpobj; /* address of object to query */
```

The OleQueryReleaseStatus function determines whether an operation has finished for the specified object.

Parameter	Description
lpobj	Points to an object for which the operation is queried.

### Returns

The return value, if the function is successful, is either OLE\_BUSY if an operation is in progress or OLE\_OK. If the pointer passed in the lpobj parameter is invalid, the function returns OLE\_ERROR\_OBJECT.

### See Also

OleQueryReleaseError, OleQueryReleaseMethod



## OleQueryServerVersion (3.1)

#include ole.h

DWORD OleQueryServerVersion(void)

The OleQueryServerVersion function retrieves the version number of the server library.

### Returns

The return value is a doubleword value. The major version number is in the low-order byte of the low-order word, and the minor version number is in the high-order byte of the low-order word. The high-order word is reserved.

### See Also

OleQueryClientVersion, HIBYTE, LOBYTE

### OleQuerySize (3.1)

#include ole.h

```
OLESTATUS OleQuerySize(lpObject, pdwSize)
```

```
LPOLEOBJECT lpObject;          /* address of object to query */
```

```
DWORD FAR* pdwSize;           /*
```

```
address of size of object
```

```
/
```

\*

The OleQuerySize function retrieves the size of the specified object.

Parameter	Description
lpObject	Points to the object to query.
pdwSize	Points to a variable for the size of the object. This variable contains the size of the object when the function returns.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_ERROR_BLANK  
OLE_ERROR_MEMORY  
OLE_ERROR_OBJECT
```

#### See Also

[OleLoadFromStream](#)

### OleQueryType (3.1)

#include ole.h

```
OLESTATUS OleQueryType(lpObject, lpType)
LPOLEOBJECT lpObject;          /* address of object to query */
LONG FAR* lpType;              /* address of type of object */
/
```

The OleQueryType function checks whether a specified object is embedded, linked, or static.

Parameter	Description								
lpObject	Points to the object for which the type is to be queried.								
lpType	Points to a long variable that contains the type of the object when the function returns. This parameter can be one of the following values:								
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>OT_EMBEDDED</td><td>Object is embedded.</td></tr><tr><td>OT_LINK</td><td>Object is a link.</td></tr><tr><td>OT_STATIC</td><td>Object is a static picture.</td></tr></tbody></table>	Value	Meaning	OT_EMBEDDED	Object is embedded.	OT_LINK	Object is a link.	OT_STATIC	Object is a static picture.
Value	Meaning								
OT_EMBEDDED	Object is embedded.								
OT_LINK	Object is a link.								
OT_STATIC	Object is a static picture.								

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_GENERIC  
OLE\_ERROR\_OBJECT

See Also

OleEnumFormats

## OleReconnect (3.1)

#include ole.h

```
OLESTATUS OleReconnect(lpObject)
LPOLEOBJECT lpObject;      /* address of object to reconnect to */
```

The OleReconnect function reestablishes a link to an open linked object. If the specified object is not open, this function does not open it.

Parameter	Description
lpObject	Points to the object to reconnect to.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_BUSY
OLE_ERROR_NOT_LINK
OLE_ERROR_OBJECT
OLE_ERROR_STATIC
OLE_WAIT_FOR_RELEASE
```

### Comments

A client application can use OleReconnect to keep the presentation for a linked object up-to-date.

### See Also

[OleActivate](#), [OleClose](#), [OleUpdate](#)

### OleRegisterClientDoc (3.1)

```
#include ole.h

OLESTATUS OleRegisterClientDoc(lpszClass, lpszDoc, reserved, lplhDoc)
LPCSTR lpszClass;          /* address of string for class name */
LPCSTR lpszDoc;           /* address of string for document name */
/* address of string for document name */
/
LONG reserved;
/* reserved */
/
LHCLIENTDOC FAR* lplhDoc;
/* address of handle of document */
/
```

The OleRegisterClientDoc function registers an open client document with the library and returns the handle of that document.

Parameter	Description
lpszClass	Points to a null-terminated string specifying the class of the client document.
lpszDoc	Points to a null-terminated string specifying the location of the client document. (This value should be a fully qualified path.)
reserved	Reserved. Must be zero.
lplhDoc	Points to the handle of the client document when the function returns. This handle is used to identify the document in other document-management functions.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_ERROR_ALREADY_REGISTERED
OLE_ERROR_MEMORY
OLE_ERROR_NAME
```

#### Comments

When a document being copied onto the clipboard exists only because the client application is copying Native data that contains objects, the name specified in the lpszDoc parameter must be Clipboard.

Client applications should register open documents with the library and notify the library when a document is renamed, closed, saved, or restored to a changed state.

#### See Also

[OleRenameClientDoc](#), [OleRevertClientDoc](#), [OleRevokeClientDoc](#), [OleSavedClientDoc](#)

### OleRegisterServerDoc (3.1)

```
#include ole.h

OLESTATUS OleRegisterServerDoc(lhsrvr, lpszDocName, lpdoc, lplhdoc)
LHSERVER lhsrvr; /* server handle */
LPCSTR lpszDocName;
/* address of string for document name
/
LPOLESERVERDOC lpdoc;
/* address of OLESERVERDOC structure
/
LHSERVERDOC FAR* lplhdoc;
/* handle of registered document
/
```

The OleRegisterServerDoc function registers a document with the server library in case other client applications have links to it. A server application uses this function when the server is started with the / Embedding filename option or when it creates or opens a document that is not requested by the library.

Parameter	Description
lhsrvr	Identifies the server. Server applications obtain this handle by calling the <u>OleRegisterServer</u> function.
lpszDocName	Points to a null-terminated string specifying the permanent name for the document. This parameter should be a fully qualified path.
lpdoc	Points to an <u>OLESERVERDOC</u> structure allocated and initialized by the server application.
lplhdoc	Points to a handle that will identify the document. This parameter points to the handle when the function returns.

#### Returns

If the function is successful, the return value is OLE\_OK. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_ADDRESS  
OLE\_ERROR\_HANDLE  
OLE\_ERROR\_MEMORY

#### Comments

If the document was created or opened in response to a request from the server library, the server should not register the document by using OleRegisterServerDoc. Instead, the server should return a pointer to the OLESERVERDOC structure through the parameter to the relevant function.

#### See Also

OleRegisterServer, OleRevokeServerDoc

## OleRegisterServer (3.1)

```
#include ole.h

OLESTATUS OleRegisterServer(lpszClass, lpsrvr, lplhserver, hinst, srvruse)
LPCSTR lpszClass;          /* address of string for class name */
LPOLESERVER lpsrvr;
/* address of OLESERVER structure
/
LHSERVER FAR* lplhserver;
/* address of server handle
/
HINSTANCE hinst;
/* instance handle
/
OLE_SERVER_USE srvruse;
/* single or multiple instances
/
```

The OleRegisterServer function registers the specified server, class name, and instance with the server library.

Parameter	Description
lpszClass	Points to a null-terminated string specifying the class name being registered.
lpsrvr	Points to an <u>OLESERVER</u> structure allocated and initialized by the server application.
lplhserver	Points to a variable of type <u>LHSERVER</u> in which the library stores the handle of the server. This handle is used in such functions as <u>OleRegisterServerDoc</u> and <u>OleRevokeServer</u> .
hinst	Identifies the instance of the server application. This handle is used to ensure that clients connect to the correct instance of a server application.
srvruse	Specifies whether the server uses a single instance or multiple instances to support multiple objects. This value must be either <u>OLE_SERVER_SINGLE</u> or <u>OLE_SERVER_MULTI</u> .

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_CLASS  
OLE\_ERROR\_MEMORY  
OLE\_ERROR\_PROTECT\_ONLY

### Comments

When the server application starts, it creates an OLESERVER structure and calls the OleRegisterServer function. Servers that support several class names can allocate a structure for each or reuse the same structure. The class name is passed to server-application functions that are called through the library, so that servers supporting more than one class can check which class is being requested.

The srvruse parameter is used when the libraries open an object. When OLE\_SERVER\_MULTI is specified for this parameter and all current instances are already editing an object, a new instance of the server is started. Servers that support the multiple document interface (MDI) typically specify OLE\_SERVER\_SINGLE.

### See Also

OleRegisterServerDoc, OleRevokeServer

## OleRelease (3.1)

#include ole.h

```
OLESTATUS OleRelease(lpObject)
LPOLEOBJECT lpObject; /* address of object to release */
```

The OleRelease function releases an object from memory and closes it if it was open. This function does not indicate that the object has been deleted from the client document.

Parameter	Description
lpObject	Points to the object to release.

### Returns

If the function is successful, the return value is OLE\_OK. Otherwise, it is an error value, which may be one of the following:

```
OLE_BUSY
OLE_ERROR_OBJECT
OLE_WAIT_FOR_RELEASE
```

### Comments

The OleRelease function should be called for all objects when closing the client document.

### See Also

OleDelete



### OleRename (3.1)

#include ole.h

```
OLESTATUS OleRename(lpObject, lpszNewname)
```

```
LPOLEOBJECT lpObject;          /* address of object being renamed */
```

```
LPCSTR lpszNewname;           /* address of string for new object name */
```

```
/* address of string for new object name
```

```
/
```

The OleRename function renames an object.

Parameter	Description
lpObject	Points to the object that is being renamed.
lpszNewname	Points to a null-terminated string specifying the new name of the object.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be OLE\_ERROR\_OBJECT.

#### Comments

Object names need not be seen by the user. They must be unique within the containing document and must be preserved when the document is saved.

#### See Also

[OleQueryName](#)

### OleRenameClientDoc (3.1)

```
#include ole.h
```

```
OLESTATUS OleRenameClientDoc(lhClientDoc, lpszNewDocname)
```

```
LHCLIENTDOC lhClientDoc;          /* handle of client document          */
```

```
LPCSTR lpszNewDocname;
```

```
/* address of string for new document name
```

```
/
```

The OleRenameClientDoc function informs the client library that a document has been renamed. A client application calls this function when a document name has changed--for example, when the user chooses the Save or Save As command from the File menu.

Parameter	Description
lhClientDoc	Identifies the document that has been renamed.
lpszNewDocname	Points to a null-terminated string specifying the new name of the document.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be OLE\_ERROR\_HANDLE.

#### Comments

Client applications should register open documents with the library and notify the library when a document is renamed, closed, saved, or restored to a changed state.

#### See Also

[OleRegisterClientDoc](#), [OleRevertClientDoc](#), [OleRevokeClientDoc](#), [OleSavedClientDoc](#)

### OleRenameServerDoc (3.1)

```
#include ole.h
```

```
OLESTATUS OleRenameServerDoc(lhDoc, lpszDocName)
```

```
LHSERVERDOC lhDoc;          /* handle of document          */
```

```
LPCSTR lpszDocName;          /* address of string for path and filename */
```

```
/
```

The OleRenameServerDoc function informs the server library that a document has been renamed.

Parameter	Description
lhDoc	Identifies the document that has been renamed.
lpszDocName	Points to a null-terminated string specifying the new name of the document. This parameter is typically a fully qualified path.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_HANDLE

OLE\_ERROR\_MEMORY

#### Comments

The OleRenameServerDoc function has the same effect as sending the OLE\_RENAMED notification to the client application's callback function. The server application calls this function when it renames a document to which the active links need to be reconnected or when the user chooses the Save As command from the File menu while working with an embedded object.

Server applications should register open documents with the server library and notify the library when a document is renamed, closed, saved, or restored to a changed state.

#### See Also

[OleRegisterServerDoc](#), [OleRevertServerDoc](#), [OleRevokeServerDoc](#), [OleSavedServerDoc](#)

### OleRequestData (3.1)

#include ole.h

```
OLESTATUS OleRequestData(lpObject, cfFormat)
LPOLEOBJECT lpObject;          /* address of object to query */
OLECLIPFORMAT cfFormat;
/* format for retrieved data
/
```

The OleRequestData function requests the library to retrieve data in a specified format from a server.

Parameter	Description
lpObject	Points to the object that is associated with the server from which data is to be retrieved.
cfFormat	Specifies the format in which data is to be returned. This parameter can be one of the predefined clipboard formats or the value returned by the <u>RegisterClipboardFormat</u> function.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_BUSY
OLE_ERROR_NOT_OPEN
OLE_ERROR_OBJECT
OLE_ERROR_STATIC
OLE_WAIT_FOR_RELEASE
```

#### Comments

The client application should be connected to the server application when the client calls the OleRequestData function. When the client receives the OLE\_RELEASE notification, it can retrieve the data from the object by using the OleGetData function or query the data by using such functions as OleQueryBounds.

If the requested data format is the same as the presentation data for the object, the library manages the data and updates the presentation.

The OleRequestData function returns OLE\_WAIT\_FOR\_RELEASE if the server is busy. In this case, the application should continue to dispatch messages until it receives a callback notification with the OLE\_RELEASE argument.

#### See Also

OleEnumFormats, OleGetData, OleSetData, RegisterClipboardFormat

### OleRevertClientDoc (3.1)

```
#include ole.h
```

```
OLESTATUS OleRevertClientDoc(lhClientDoc)
```

```
LHCLIENTDOC lhClientDoc;          /* handle of client document */
```

The OleRevertClientDoc function informs the library that a document has been restored to a previously saved condition.

Parameter	Description
lhClientDoc	Identifies the document that has been restored to its saved state.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be OLE\_ERROR\_HANDLE.

#### Comments

A client application should call the OleRevertClientDoc function when it reloads a document without saving changes to the document.

Client applications should register open documents with the library and notify the library when a document is renamed, closed, saved, or restored to a saved state.

#### See Also

[OleRegisterClientDoc](#), [OleRenameClientDoc](#), [OleRevokeClientDoc](#), [OleSavedClientDoc](#)

### OleRevertServerDoc (3.1)

```
#include ole.h
```

```
OLESTATUS OleRevertServerDoc(lhDoc)  
LHSERVERDOC lhDoc;          /* handle of document */
```

The OleRevertServerDoc function informs the server library that the server has restored a document to its saved state without closing it.

Parameter	Description
lhDoc	Identifies the document that has been restored to its saved state.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be OLE\_ERROR\_HANDLE.

#### Comments

Server applications should register open documents with the server library and notify the library when a document is renamed, closed, saved, or restored to a saved state.

#### See Also

[OleRegisterServerDoc](#), [OleRenameServerDoc](#), [OleRevokeServerDoc](#), [OleSavedServerDoc](#)

## OleRevokeClientDoc (3.1)

```
#include ole.h
```

```
OLESTATUS OleRevokeClientDoc(lhClientDoc)  
LHCLIENTDOC lhClientDoc;          /* handle of client document */
```

The OleRevokeClientDoc function informs the client library that a document is no longer open.

Parameter	Description
lhClientDoc	Identifies the document that is no longer open. This handle is invalid following the call to OleRevokeClientDoc.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_ERROR_HANDLE  
OLE_ERROR_NOT_EMPTY
```

### Comments

The client application should delete all the objects in a document before calling OleRevokeClientDoc.

Client applications should register open documents with the library and notify the library when a document is renamed, closed, saved, or restored to a changed state.

### See Also

[OleRegisterClientDoc](#), [OleRenameClientDoc](#), [OleRevertClientDoc](#), [OleSavedClientDoc](#)

## OleRevokeServerDoc (3.1)

#include ole.h

```
OLESTATUS OleRevokeServerDoc(lhdoc)
LHSERVERDOC lhdoc;          /* document handle */
```

The OleRevokeServerDoc function revokes the specified document. A server application calls this function when a registered document is being closed or otherwise made unavailable to client applications.

Parameter	Description
lhdoc	Identifies the document to revoke. This handle was returned by a call to the <u>OleRegisterServerDoc</u> function or was associated with a document by using one of the server-supplied functions that create documents.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_HANDLE  
OLE\_WAIT\_FOR\_RELEASE

### Comments

If this function returns OLE\_WAIT\_FOR\_RELEASE, the server application should not free the OLESERVERDOC structure or exit until the library calls the server's Release function.

### See Also

OleRegisterServerDoc, OleRevokeObject, OleRevokeServer, OLESERVERDOC



## OleRevokeObject (3.1)

#include ole.h

```
OLESTATUS OleRevokeObject(lpClient)
LPOLECLIENT lpClient;          /* address of OLECLIENT structure */
```

The OleRevokeObject function revokes access to an object. A server application typically calls this function when the user destroys an object.

Parameter	Description
lpClient	Points to the <u>OLECLIENT</u> structure associated with the object being revoked.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### See Also

OleRevokeServer, OleRevokeServerDoc

## OleRevokeServer (3.1)

#include ole.h

```
OLESTATUS OleRevokeServer(lhServer)
LHSERVER lhServer;          /* server handle */
```

The OleRevokeServer function is called by a server application to close any registered documents.

Parameter	Description
lhServer	Identifies the server to revoke. A server application obtains this handle in a call to the <u>OleRegisterServer</u> function.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_HANDLE  
OLE\_WAIT\_FOR\_RELEASE

### Comments

The OleRevokeServer function returns OLE\_WAIT\_FOR\_RELEASE if communications between clients and the server are in the process of terminating. In this case, the server application should continue to send and dispatch messages until the library calls the server's Release function.

### See Also

OleRegisterServer, OleRevokeObject, OleRevokeServerDoc

### OleSavedClientDoc (3.1)

#include ole.h

```
OLESTATUS OleSavedClientDoc(lhClientDoc)
LHCLIENTDOC lhClientDoc;          /* handle of client document */
```

The OleSavedClientDoc function informs the client library that a document has been saved.

Parameter	Description
lhClientDoc	Identifies the document that has been saved.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be OLE\_ERROR\_HANDLE.

#### Comments

Client applications should register open documents with the client library and notify the library when a document is renamed, closed, saved, or restored to a saved state.

#### See Also

[OleRegisterClientDoc](#), [OleRenameClientDoc](#), [OleRevertClientDoc](#), [OleRevokeClientDoc](#)

## OleSavedServerDoc (3.1)

```
#include ole.h  
OLESTATUS OleSavedServerDoc(lhDoc)  
LHSERVERDOC lhDoc;          /* handle of document */
```

The OleSavedServerDoc function informs the server library that a document has been saved.

Parameter	Description
lhDoc	Identifies the document that has been saved.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_ERROR_CANT_UPDATE_CLIENT  
OLE_ERROR_HANDLE
```

### Comments

The OleSavedServerDoc function has the same effect as sending the OLE\_SAVED notification to the client application's callback function. The server application calls this function when saving a document or when updating an embedded object without closing the document.

When a server application receives the OLE\_ERROR\_CANT\_UPDATE\_CLIENT error value, it should display a message box indicating that the user cannot update the document until the server terminates.

Server applications should register open documents with the server library and notify the library when a document is renamed, closed, saved, or restored to a saved state.

### See Also

[OleRegisterServerDoc](#), [OleRenameServerDoc](#), [OleRevertServerDoc](#), [OleRevokeServerDoc](#)

## OleSaveToStream (3.1)

#include ole.h

```
OLESTATUS OleSaveToStream(lpObject, lpStream)
```

```
LPOLEOBJECT lpObject; /* address of object to save */
```

```
LPOLESTREAM lpStream;
```

```
/* address of OLESTREAM structure
```

```
/
```

The OleSaveToStream function saves an object to the stream.

Parameter	Description
lpObject	Points to the object to be saved to the stream.
lpStream	Points to an <u>OLESTREAM</u> structure allocated and initialized by the client application. The library calls the <u>Put</u> function in the <u>OLESTREAM</u> structure to store the data from the object.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_ERROR_BLANK  
OLE_ERROR_MEMORY  
OLE_ERROR_OBJECT  
OLE_ERROR_STREAM
```

### Comments

An application can use the OleQuerySize function to find the number of bytes to allocate for the object.

### See Also

OleLoadFromStream, OleQuerySize

## OleSetBounds (3.1)

#include ole.h

```
OLESTATUS OleSetBounds(lpObject, lprcBound)
LPOLEOBJECT lpObject;          /* address of object          */
RECT FAR* lprcBound;          /* address of structure for bounding rectangle */
/
```

The OleSetBounds function sets the coordinates of the bounding rectangle for the specified object on the target device.

Parameter	Description
lpObject	Points to the object for which the bounding rectangle is set.
lprcBound	Points to a <u>RECT</u> structure containing the coordinates of the bounding rectangle. The coordinates are specified in MM_HIMETRIC units. Neither the width nor height of an object should exceed 32,767 MM_HIMETRIC units.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_BUSY  
OLE\_ERROR\_MEMORY  
OLE\_ERROR\_OBJECT  
OLE\_WAIT\_FOR\_RELEASE

The OleSetBounds function returns OLE\_ERROR\_OBJECT when it is called for a linked object.

### Comments

The OleSetBounds function is ignored for linked objects, because the size of a linked object is determined by the source document for the link.

A client application uses OleSetBounds to change the bounding rectangle. The client does not need to call OleSetBounds every time a server is opened.

The bounding rectangle specified in the OleSetBounds function does not necessarily have the same dimensions as the rectangle specified in the call to the OleDraw function. These dimensions may be different because of the view scaling used by the container application. An application can use OleSetBounds to cause the server to reformat the picture to fit the rectangle more closely.

In the MM\_HIMETRIC mapping mode, the positive y-direction is up.

### See Also

OleDraw, OleQueryBounds, SetMapMode, RECT

## OleSetColorScheme (3.1)

```
#include ole.h
OLESTATUS OleSetColorScheme(lpObject, lpPalette)
LPOLEOBJECT lpObject;          /* address of object          */
const LOGPALETTE FAR* lpPalette; /* address of preferred palette */
/
```

The OleSetColorScheme function specifies the palette a client application recommends be used when the server application edits the specified object. The server application can ignore the recommended palette.

Parameter	Description
lpObject	Points to an <u>OLEOBJECT</u> structure describing the object for which a palette is recommended.
lpPalette	Points to a <u>LOGPALETTE</u> structure specifying the recommended palette.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_BUSY
OLE_ERROR_COMM
OLE_ERROR_MEMORY
OLE_ERROR_OBJECT
OLE_ERROR_PALETTE
OLE_ERROR_STATIC
OLE_WAIT_FOR_RELEASE
```

The OleSetColorScheme function returns OLE\_ERROR\_OBJECT when it is called for a linked object.

### Comments

A client application uses OleSetColorScheme to change the color scheme. The client does not need to call OleSetColorScheme every time a server is opened.

The first palette entry in the LOGPALETTE structure specifies the foreground color recommended by the client application. The second palette entry specifies the background color. The first half of the remaining palette entries are fill colors, and the second half are colors for lines and text.

Client applications should specify an even number of palette entries. When there is an uneven number of entries, the server interprets the odd entry as a fill color; that is, if there are five entries, three are interpreted as fill colors and two as line and text colors.

When server applications render metafiles, they should use the suggested palette.

### See Also

LOGPALETTE

### OleSetData (3.1)

```
#include ole.h

OLESTATUS OleSetData(lpObject, cfFormat, hData)
LPOLEOBJECT lpObject;          /* address of object */
OLECLIPFORMAT cfFormat;      /
/* format of data to send
/
HANDLE hData;
/* memory containing data
/
```

The OleSetData function sends data in the specified format to the server associated with a specified object.

Parameter	Description
lpObject	Points to an object specifying the server to which data is to be sent.
cfFormat	Specifies the format of the data.
hData	Identifies a memory object containing the data in the specified format.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_BUSY
OLE_ERROR_BLANK
OLE_ERROR_MEMORY
OLE_ERROR_NOT_OPEN
OLE_ERROR_OBJECT
OLE_WAIT_FOR_RELEASE
```

If the specified object cannot accept the data, the function returns an error value. If the server is not open and the requested data format is different from the format of the presentation data, the return value is OLE\_ERROR\_NOT\_OPEN.

#### See Also

[OleGetData](#), [OleRequestData](#)



### OleSetHostNames (3.1)

```
#include ole.h
```

```
OLESTATUS OleSetHostNames(lpObject, lpszClient, lpszClientObj)
```

```
LPOLEOBJECT lpObject; /* address of object */
```

```
*/
```

```
LPCSTR lpszClient;
```

```
/* address of string with name of client app
```

```
/
```

```
LPCSTR lpszClientObj;
```

```
/* address of string with client's name for object
```

```
/
```

The OleSetHostNames function specifies the name of the client application and the client's name for the specified object. This information is used in window titles when the object is being edited in the server application.

Parameter	Description
lpObject	Points to the object for which a name is to be set.
lpszClient	Points to a null-terminated string specifying the name of the client application.
lpszClientObj	Points to a null-terminated string specifying the client's name for the object.

#### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_BUSY

OLE\_ERROR\_MEMORY

OLE\_ERROR\_OBJECT

OLE\_WAIT\_FOR\_RELEASE

The OleSetHostNames function returns OLE\_ERROR\_OBJECT when it is called for a linked object.

#### Comments

When a server application is started for editing of an embedded object, it displays in its title bar the string specified in the lpszClientObj parameter. The object name specified in this string should be the name of the client document containing the object.

A client application uses OleSetHostNames to set the name of an object the first time that object is activated or to change the name of an object. The client does not need to call OleSetHostNames every time a server is opened.

## OleSetLinkUpdateOptions (3.1)

#include ole.h

```
OLESTATUS OleSetLinkUpdateOptions(lpObject, UpdateOpt)
LPOLEOBJECT lpObject;          /* address of object */
OLEOPT_UPDATE UpdateOpt;
/* link-update options
/
```

The OleSetLinkUpdateOptions function sets the link-update options for the presentation of the specified object.

Parameter	Description
lpObject	Points to the object for which the link-update option is set.
UpdateOpt	Specifies the link-update option for the specified object. This parameter can be one of the following values:

Option	Description
oleupdate_always	Update the linked object whenever possible. This option supports the Automatic link-update radio button in the Links dialog box.
oleupdate_oncall	Update the linked object only on request from the client application. This option supports the Manual link-update radio button in the Links dialog box.
oleupdate_onsave	Update the linked object when the source document is saved by the server.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_BUSY  
OLE\_ERROR\_OBJECT  
OLE\_ERROR\_OPTION  
OLE\_ERROR\_STATIC  
OLE\_WAIT\_FOR\_RELEASE

### See Also

[OleGetLinkUpdateOptions](#)

## OleSetTargetDevice (3.1)

#include ole.h

```
OLESTATUS OleSetTargetDevice(lpObject, hotd)
LPOLEOBJECT lpObject;          /* address of object */
HGLOBAL hotd;                  /
/* handle of OLETARGETDEVICE structure
/
```

The OleSetTargetDevice function specifies the target output device for an object.

Parameter	Description
lpObject	Points to the object for which a target device is specified.
hotd	Identifies an <u>OLETARGETDEVICE</u> structure that describes the target device for the object.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_BUSY
OLE_ERROR_MEMORY
OLE_ERROR_OBJECT
OLE_ERROR_STATIC
OLE_WAIT_FOR_RELEASE
```

### Comments

The OleSetTargetDevice function allows a linked or embedded object to be formatted correctly for a target device, even when the object is rendered on a different device. A client application should call this function whenever the target device changes, so that servers can be notified to change the rendering of the object, if necessary. The client application should call the OleUpdate function to ensure that the information is sent to the server, so that the server can make the necessary changes to the object's presentation. The client application should call the library to redraw the object if it receives a notification from the server that the object has changed.

A client application uses the OleSetTargetDevice function to change the target device. The client does not need to call OleSetTargetDevice every time a server is opened.

### See Also

OLETARGETDEVICE

## OleUnblockServer (3.1)

#include ole.h

```
OLESTATUS OleUnblockServer(lhSrvr, lpfRequest)
```

```
LHSERVER lhSrvr; /* handle of server */
```

```
BOOL FAR* lpfRequest;
```

```
/* address of flag for more requests
```

```
 /
```

The OleUnblockServer function processes a request from a queue created by calling the OleBlockServer function.

Parameter	Description
lhSrvr	Identifies the server for which requests were queued.
lpfRequest	Points to a flag indicating whether there are further requests in the queue. If there are further requests in the queue, this flag is TRUE when the function returns. Otherwise, it is FALSE when the function returns.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_HANDLE  
OLE\_ERROR\_MEMORY

### Comments

A server application can use the OleBlockServer and OleUnblockServer functions to control when the server library processes requests from client applications. It is best to use OleUnblockServer outside the GetMessage function in a message loop, unblocking all blocked messages before getting the next message. Unblocking message loops should not be run inside server-defined functions that are called by the library.

### See Also

OleBlockServer

## OleUnlockServer (3.1)

#include ole.h

```
OLESTATUS OleUnlockServer(hServer)
LHSERVER hServer;          /* handle of server to unlock */
```

The OleUnlockServer function unlocks a server that was locked by the OleLockServer function.

Parameter	Description
hServer	Identifies the server to release from memory. This handle was retrieved by a call to the <u>OleLockServer</u> function.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_ERROR\_HANDLE  
OLE\_WAIT\_FOR\_RELEASE

### Comments

When the OleLockServer function is called more than once for a given server, the server's lock count is incremented. Each call to OleUnlockServer decrements the lock count. The server remains locked until the lock count is zero.

If the OleUnlockServer function returns OLE\_WAIT\_FOR\_RELEASE, the application should call the OleQueryReleaseStatus function to determine whether the unlocking process has finished. In the call to OleQueryReleaseStatus, the application can cast the server handle to a long pointer to an object linking and embedding (OLE) object (LPOLEOBJECT):

```
OleQueryReleaseStatus((LPOLEOBJECT) lhserver);
```

When OleQueryReleaseStatus no longer returns OLE\_BUSY, the server has been unlocked.

### See Also

OleLockServer, OleQueryReleaseStatus

## OleUpdate (3.1)

#include ole.h

```
OLESTATUS OleUpdate(lpObject)
LPOLEOBJECT lpObject; /* address of object */
```

The OleUpdate function updates the specified object. This function updates the presentation of the object and ensures that the object is up-to-date with respect to any linked objects it contains.

Parameter	Description
lpObject	Points to the object to be updated.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

OLE\_BUSY  
OLE\_ERROR\_OBJECT  
OLE\_ERROR\_STATIC  
OLE\_WAIT\_FOR\_RELEASE

### See Also

[OleQueryOutOfDate](#)

### OLE functions (3.1)

<u>OleActivate</u>	Activates an object
<u>OleBlockServer</u>	Queues incoming requests for the server
<u>OleClone</u>	Makes a copy of an object
<u>OleClose</u>	Closes a specified open object
<u>OleCopyFromLink</u>	Makes an embedded copy of a linked object
<u>OleCopyToClipboard</u>	Puts the specified object on the clipboard
<u>OleCreate</u>	Creates an embedded object of a specified class
<u>OleCreateFromClip</u>	Creates an object from the clipboard
<u>OleCreateFromFile</u>	Creates an embedded object from a file
<u>OleCreateFromTemplate</u>	Creates an object from a template
<u>OleCreateInvisible</u>	Creates an object without displaying it
<u>OleCreateLinkFromClip</u>	Creates link to object from the clipboard
<u>OleCreateLinkFromFile</u>	Creates a link to an object in a file
<u>OleDelete</u>	Deletes an object and frees associated memory
<u>OleDraw</u>	Draws a specified object into a device context
<u>OleEnumFormats</u>	Enumerates data formats for an object
<u>OleEnumObjects</u>	Enumerates objects in a document
<u>OleEqual</u>	Compares two objects for equality
<u>OleExecute</u>	Sends DDE execute commands to a server
<u>OleGetData</u>	Retrieves data from an object in a specified format
<u>OleGetLinkUpdateOptions</u>	Retrieves link-update options for an object
<u>OleIsDcMeta</u>	Identifies a metafile device context
<u>OleLoadFromStream</u>	Loads an object from the containing document
<u>OleLockServer</u>	Keeps an open server application in memory
<u>OleObjectConvert</u>	Creates a new object using a specified protocol
<u>OleQueryBounds</u>	Retrieves a bounding rectangle for the object
<u>OleQueryClientVersion</u>	Retrieves the version number of a client library
<u>OleQueryCreateFromClip</u>	Retrieves presentation data for a clipboard object
<u>OleQueryLinkFromClip</u>	Retrieves link data for a clipboard object
<u>OleQueryName</u>	Retrieves the name of an object
<u>OleQueryOpen</u>	Determines whether an object is open
<u>OleQueryOutOfDate</u>	Determines whether an object is out-of-date
<u>OleQueryProtocol</u>	Determines whether an object supports a protocol
<u>OleQueryReleaseError</u>	Determines the status of a released operation
<u>OleQueryReleaseMethod</u>	Determines which server operation released
<u>OleQueryReleaseStatus</u>	Determines whether an operation released
<u>OleQueryServerVersion</u>	Retrieves the version number of a server library
<u>OleQuerySize</u>	Retrieves the size of an object
<u>OleQueryType</u>	Checks if object is linked, embedded, or static
<u>OleReconnect</u>	Reconnects to an open linked object
<u>OleRegisterClientDoc</u>	Registers an open client document with the library
<u>OleRegisterServerDoc</u>	Registers a document with the server library
<u>OleRegisterServer</u>	Registers the specified server
<u>OleRelease</u>	Releases an object from memory and closes it
<u>OleRename</u>	Informs library that an object is renamed
<u>OleRenameClientDoc</u>	Informs library that a document is renamed
<u>OleRenameServerDoc</u>	Informs library that a document is renamed
<u>OleRequestData</u>	Retrieves data from a server in a specified format
<u>OleRevertClientDoc</u>	Informs library that a doc reverted to saved state
<u>OleRevertServerDoc</u>	Informs library that a doc is reset to saved state
<u>OleRevokeClientDoc</u>	Informs library that a document is not open
<u>OleRevokeServerDoc</u>	Revokes the specified document
<u>OleRevokeObject</u>	Revokes access to an object
<u>OleRevokeServer</u>	Revokes the specified server
<u>OleSavedClientDoc</u>	Informs library that a document has been saved
<u>OleSavedServerDoc</u>	Informs library that a document has been saved
<u>OleSaveToStream</u>	Saves an object to the stream
<u>OleSetBounds</u>	Sets a bounding rectangle for the object
<u>OleSetColorScheme</u>	Specifies a client's recommended object colors
<u>OleSetData</u>	Sends data in specified format to server
<u>OleSetHostNames</u>	Sets the client name and object name for a server
<u>OleSetLinkUpdateOptions</u>	Sets link-update options for an object
<u>OleSetTargetDevice</u>	Sets target output device for an object

OleUnblockServer  
OleUnlockServer  
OleUpdate

Processes requests from a queue  
Releases a server locked with OleLockServer  
Updates an object



## Windows Overviews (3.1)

32-Bit Memory Management Library

Common Dialog Box Library

Compatibility Issues

Control Panel Applications

Creating Windows Applications

Data Decompression Library

DDE Management Library

DPML Applications

File Formats

File Installation

File Manager Extensions

Floating Point Emulation Library

Fonts

Graphics Device Interface

Installable Drivers

International Applications

Network Applications

Object Linking and Embedding

Screen Saver Library

Self-Loading Windows Applications

Shell Dynamic Data Exchange Interface

Shell Library

Stress-Testing Library

Tool Helper Library

Video Techniques

Window Management

Windows Application Startup

Windows Debugging Version

Windows Prologs and Epilogs

File Formats (3.1)  
Calendar File Format  
Clipboard-File Format  
Executable-File Format  
Font-File Format  
Graphics-File Formats  
Group-File Format  
Metafile Format  
Resource Formats  
Symbol File Format  
Write File Format

## Video Techniques

This topic describes some techniques that can improve the video performance of applications for the Microsoft Windows operating system. These techniques include using an identity palette to speed up image drawing, accommodating differences in video adapters, and modifying device-independent bitmaps (DIBs) by using the DIB driver.

### Using an Identity Palette

Windows reserves a group of system palette entries for a fixed number of colors. These colors, which are named system colors, are used for drawing screen elements such as scroll bars. Windows also uses the system colors as replacement color entries when inactive windows request more color entries than are available in the system palette. Windows places the system colors at the top and bottom of the system palette to ensure that logical operations (such as XOR) work correctly.

By arranging logical palettes the same way that Windows arranges the system palette, you can avoid unexpected color changes and improve the speed at which your application draws DIBs. To do this, you must create an identity palette, a logical palette that matches the system palette. To use identity palettes, however, you need to understand how Windows sets up the system palette.

### Understanding the System Palette

When an application realizes a palette (that is, requests the palette be given specified colors), Windows adds the logical palette entries to the system palette. Windows always reserves system palette entries for the system colors. For example, a 256-color video graphics adapter (VGA) driver with 20 system colors allows an application to use a maximum of 236 system palette entries. If a logical palette contains more entries than can fit in the system palette (after the system colors are added), Windows truncates the palette, using only as many colors as it can fit without encroaching on the reserved system colors. You can force Windows to relinquish the system color entries (by using the `SetSystemPaletteUse` function), but by doing so you change the coloring of all Windows screen elements to black and white.

The maximum number of colors available to a foreground window equals the number of colors supported by the video driver minus the number of system reserved colors and the number of palette entries reserved by the application.

Windows places the system colors at the top and bottom of the system palette. For example, a 256-color VGA driver uses the top 10 and bottom 10 system palette entries for the system colors. If a logical palette does not contain the system colors or if the system colors appear in locations other than the default positions, Windows changes the ordering of the palette entries when your application realizes its palette. At this point, logical palette entry *n* does not necessarily match system palette entry *n*. When your application draws a bitmap to the device context, Windows must translate the bitmap palette indices to the new locations on the system palette. This translation step takes time.

The goal is to make the logical palette exactly match the system palette. By doing so, your images can be colored exactly as you expect. The video driver can also draw the images faster because the translation step is avoided.

### Creating an Identity Palette

An identity palette is a logical palette that exactly matches the system palette and therefore has the same number of entries as the system palette and includes color entries for the system colors. The system colors appear at the top and bottom of the color table.

The Microsoft Windows Paintbrush application always saves bitmaps with an identity palette. To convert a bitmap palette to an identity palette, you can open the bitmap in Paintbrush and then save it.

### Accommodating Different Video Adapters and Drivers

This section contains information on adapting your logical palette to a different display type.

#### Distinguishing Between Standard VGA and Super VGA

Most super VGA adapters are single-plane devices, which makes them well-suited for displaying DIBs. On a super VGA adapter, there is little speed difference between drawing DIBs and drawing device-dependent bitmaps--you can choose whichever format is more convenient for your application.

Standard VGA adapters have multiple planes and are not as well suited for displaying DIBs. It is faster to work with device-dependent bitmaps on standard VGA. To determine whether a standard VGA adapter is present, use the following code:

```
hDC = CreateDC("DISPLAY", NULL, NULL, NULL);
```

```
bIsMultiplane = (GetDeviceCaps(hdc, PLANES) > 1);
```

```
DeletedDC(hdc);
```

#### Adapting Identity Palettes to Different Display Adapters

Even if two display devices use the same number of system colors, you cannot assume that the red, green, and blue (RGB) values for the low-intensity colors match. One particular problem is the difference between super VGA and 8514 systems. Both provide 256 colors and use 20 system colors, but the low-intensity system color values for the VGA are different from those for the 8514. An identity palette created on a VGA system is not the same as an identity palette on an 8514 system.

If you create an identity palette on a VGA system and then display the DIB on an 8514 system, Windows recognizes the low-intensity colors in the logical palette as custom colors rather than system colors. It puts these colors in the custom-color section of the palette (in entries 10 through 245) and the 8514 system colors in the top and bottom of the system palette.

To avoid misrecognition of colors, an application can do the following:

- 1 When the application loads, it should use the `GetSystemColors` function to retrieve the system colors from the system palette and compare these colors against the system colors used in the DIB palettes.
- 2 If the colors do not match, the application should copy the current system colors (retrieved from the system palette) over the DIB system colors.

#### Using a Device-Independent Bitmap Driver

Many MS-DOS applications manipulate screen memory directly. To maintain the device independence of Windows, it is not possible to allow an application to access screen memory directly. However, an application can use the DIB driver (`DIB.DRV`) to directly manipulate an image in memory.

#### Creating a Driver Display Context

An application can load the DIB driver by passing the DIB driver name and a `BITMAPINFO` structure containing the DIB bits to the `CreateDC` function. For example, the following example creates a DIB display context that represents the packed DIB described by the `BITMAPINFO` structure `bi`:

```
hdc = CreateDC("DIB", NULL, NULL, &bi);
```

An application must observe the following rules when working with a device context created in this manner:

- If the last parameter of `CreateDC` is `NULL`, the display context is associated with a 0-by-0 8-bit DIB. Any attempt to draw with it will fail.
- The `BITMAPINFO` structure must remain locked for the life of the device context.
- The DIB driver supports 1-bit, 4-bit, or 8-bit DIB bitmaps. The run-length encoding (RLE) format is not supported.
- The DIB driver supports only Windows version 3.0 or later DIB headers.
- Multiple DIB-driver display contexts can be active.
- DIBs reside in the memory-based image buffer in the `CF_DIB` (packed-DIB) format.
- The DIB driver expects the RGBQUAD structure for color matching; it does not use palette indices. (If an application uses an RGB value for drawing, the DIB driver uses the closest match found in the color table of the DIB.)

The following example uses the DIB driver to draw a circle in a DIB copied from the clipboard:

```
if (IsClipboardFormatAvailable(CF_DIB) && OpenClipboard()) {
    HANDLE hdir;
    HDC hdc;

    /* Get the DIB from the clipboard. */

    hdir = GetClipboardData(CF_DIB);

    /* Create a DIB driver hdc on the DIB surface. */

    hdc = CreateDC("DIB", NULL, NULL,
(LPBITMAPINFO) GlobalLock(hdir));

    /* Draw a circle in the DIB. */
```

```

Ellipse(hdc, 0, 0, 100, 100);

/* Delete the DIB driver HDC now that you are done with it. */
DeleteDC(hdc);

/* Unlock the DIB. */
GlobalUnlock(hdib);

/* Release the clipboard. */
CloseClipboard();
}

```

### Moving Bitmaps to and from the Display

The DIB driver is a separate driver and is not associated with the display driver. Because of this, an application cannot use the BitBlt function to move bitmaps between a DIB-driver device context and a screen device context. An application can use the GetDIBits function to copy from the screen device context to a DIB device context. To copy a DIB device context to the screen device context, an application can use the StretchDIBits function.

An application can maximize the speed of StretchDIBits by using one of the following methods:

- One-to-one mapping for the palette
- DIB\_PAL\_COLORS, an option that prevents color matching by the graphics device interface (GDI)

### Modifying Bitmaps

DIBs offer many advantages over device-dependent bitmaps. Unlike device-dependent bitmaps, however, DIBs cannot be selected into a video device context. Before the DIB driver was available, this meant that applications could not take advantage of the extensive graphics device interface (GDI) functions to modify DIBs directly. To use GDI routines to draw in or otherwise modify a DIB, an application would follow a procedure such as this:

- 1 Create a memory device context.
- 2 Use the CreateDIBitmap function to convert the DIB to device-dependent format.
- 3 Select the device-dependent bitmap into the memory device context.
- 4 Call GDI routines to modify the device-dependent bitmap.
- 5 Use the GetDIBits function to convert the device-dependent bitmap to DIB format.

This method works well if you only use GDI routines to modify the bitmap. If you want to speed up certain operations by writing replacement functions that directly modify the DIB bits, however, the procedure can become complicated. The direct-manipulation routines work on the DIB, but the GDI routines work on the device-dependent bitmap.

Direct manipulation can be considerably faster than using equivalent GDI routines; in one sample application, a direct-manipulation function (drawing a triangle) ran eight times faster than the equivalent GDI operation. Also, direct-manipulation routines for other products may be reusable.

The DIB driver makes it possible for you to mix GDI calls with direct-manipulation routines, so you can combine the advantages of both methods.

### Creating a Driver Device Context

The DIB driver makes it possible for you to create a DIB device context. To create the DIB device context, call the CreateDC function, supplying a pointer to a BITMAPINFO structure:

```
hdc = CreateDC("DIB", NULL, NULL, lpbi);
```

You can use the device-context handle returned by the CreateDC function with most GDI functions to modify the bitmap. Concurrently, you can call your own direct-manipulation functions to modify the actual bitmap bits. Any changes made directly to the bitmap bits are reflected in the DIB-driver device context. When you finish modifying the bitmap, you can use the StretchDIBits function to transfer the DIB to the video device context.

The DIB driver can handle 1-bit, 4-bit, or 8-bit DIBs. You can create multiple DIB driver contexts. Note the following limitations:

- 1 The BITMAPINFO structure must be locked for the life of the device context.
- 2 The DIB driver handles only the Windows BITMAPINFOHEADER format.
- 3 The RLE format is not supported.
- 4 The DIB must use the DIB\_RGB\_COLORS format. The DIB driver does not support the DIB\_PAL\_COLORS (palette indexes) format.

You can distribute the DIB driver with applications that run under Windows.

## Fonts Overview

This topic describes the fonts an application can use with the Microsoft Windows 3.1 operating system and discusses how to use Windows font functions in applications. The information includes a description of TrueType font technology, which is new for Windows 3.1.

The following topics discuss the use of fonts in Windows applications:

Font Fundamentals

Fonts in Windows

TrueType Font Technology

Using Fonts in Applications

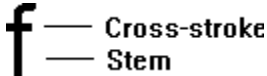
## Font Fundamentals

The vocabulary used to describe fonts may be unfamiliar to application developers. This section defines some of the terms and concepts that a developer needs to use when describing a font.

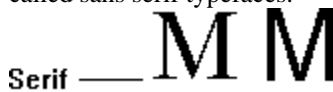
### Font Organization

A typeface is a collection of characters that share design characteristics; for example, Courier is a common typeface. A font is a collection of characters that have the same typeface and size.

The Windows graphics device interface (GDI) organizes fonts by family; each family consists of fonts that have a common design. Families are distinguished by stroke width and serif characteristics. A stroke is a horizontal or vertical line. A horizontal stroke is called a cross-stroke. The main vertical line in a character is called a stem.



Serifs are short cross-lines drawn at the ends of the main strokes of a letter. Typefaces without serifs are called sans serif typefaces.



Within a font family, fonts are distinguished by stylistic variations that generally involve their weight and slant. Weights are described by adjectives such as "extra light," "light," "demi," "demi bold," "book," "bold," "heavy bold," "extra bold," and "black." The slant of a font is described by "roman," "italic," and "oblique." A roman font is the upright form of the font; an oblique font is slanted; and an italic font is both slanted and relatively cursive. Font families usually do not include both italic and oblique fonts.

GDI uses five family names to categorize typefaces and fonts. A sixth name (FF\_DONTCARE) allows an application to use the default font. Following are the font-family names, each described briefly:

Font-family name	Description
FF_DECORATIVE	Specifies a novelty font. An example is Old English.
FF_DONTCARE	Specifies a generic family name. This name is used when information about a font does not exist or does not matter.
FF_MODERN	Specifies a font that has a constant stroke width, with or without serifs. Fixed-pitch fonts are usually modern; examples include Pica, Elite, and Courier New®.
FF_ROMAN	Specifies a font that has a variable stroke width, with serifs. An example is Times New Roman®.
FF_SCRIPT	Specifies a font that is designed to look like handwriting; examples include Script and Cursive.
FF_SWISS	Specifies a font that has a variable stroke width, without serifs. An example is Arial®.

GDI family names do not always correspond to traditional typographic categories.

### Measuring Characters

Both the visible and invisible parts of a character affect its measurement. The visible part of a character is called a glyph. The invisible part is a rectangular region that contains the character; this region is called a character cell. The origin of a character cell is its upper-left corner. When a text-output function specifies coordinates at which the text should appear, GDI places the origin of the first character cell at those coordinates. (This is the default behavior for GDI. An application can change this at any time by using the [SetTextAlign](#) function.)

The most common unit of measurement for measuring characters is the point. In the computer industry, a point is exactly 1/72 of an inch. Font heights in Windows can be specified in "twips," which are 1/20 of a point (that is, 1/1440 of an inch). Point size refers to the size of the character cell, but only loosely to the size of the visible characters; the glyphs from different 12-point fonts can have different heights.

The following example shows the different font heights in alternating glyphs from Courier New, Times New Roman, and Arial at 18 points:

**AAAaaaBBBbbbCCCcccDDDddd**



Following are some of the character-cell measurements an application can affect or query when it creates a font:

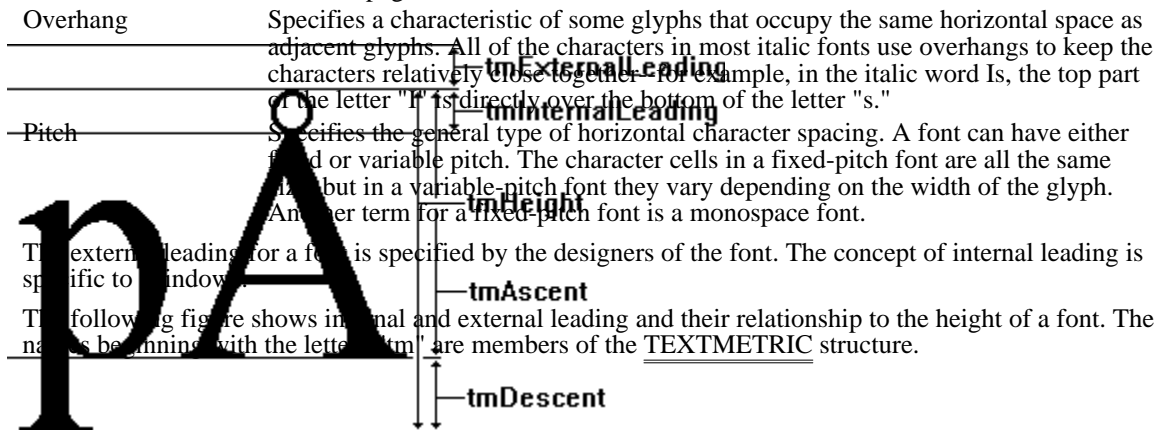
Measurement	Description
Ascent	Specifies the distance from the base line to the top of a character. The ascender of a character is the part of the character above the base line. In Windows, the value for the ascent is the distance from the base line to the top of the character cell; this can include white space. The typographic ascent, on the other hand, corresponds to the tallest character in a font. For TrueType fonts, this character is often the lowercase "f."
Base line	Specifies the line on which all characters stand. The base line is typically the lowest point of most of the capital letters in a font. (Though the tail of the "Q," for example, can extend below the base line.)
Descent	Specifies the distance from the base line to the bottom of a character. The descender of a character is the part of the character below the base line. For example, the tail of the letter "g" is a descender. In Windows, the value for the descent is the distance from the base line to the bottom of the character cell; this can include white space. The typographic descent, on the other hand, corresponds to the character in a font that extends farthest beneath the base line. For TrueType fonts, this character is often the lowercase "g."
Height	Specifies the vertical space required for a font. The height of a font is the sum of the ascent, descent, and internal leading for that font.
Width	Specifies the horizontal space required for a character cell in a font. GDI returns widths for the average character cell in a font and for the widest character cell. The average width can be simple or weighted, depending on the font. An application can also retrieve the widths for individual characters. These widths include the empty space preceding and following the glyph.

#### Measuring Line and Intercharacter Spacing

Line spacing, like character size, is typically specified in points. If a 10-point font is displayed with 12-point line spacing, this is abbreviated as "10/12" and is called "ten on twelve" line spacing.

Following are some of the line and intercharacter measurements an application can affect or query when it creates a font:

Measurement	Description
External leading	Specifies the space between rows of text. External leading is not part of the character cell. When the internal leading for a font does not contain parts of characters, the apparent line spacing is the external leading plus the internal leading. Windows does not support negative values for external leading.
Internal leading	Specifies the difference between the height of the character glyphs for a font (the font's em square) and the height of the character cell for a font. Applications use internal leading to determine the point size for a font; the point size is the height of the character cell minus its internal leading. Some applications have used internal leading incorrectly; specifically, internal leading is not strictly reserved for diacritical marks, nor should it be used as the space to be removed from the first line on a page.
Overhang	Specifies a characteristic of some glyphs that occupy the same horizontal space as adjacent glyphs. All of the characters in most italic fonts use overhangs to keep the characters relatively close together—for example, in the italic word <i>Is</i> , the top part of the letter "I" is directly over the bottom of the letter "s."



The external leading for a font is specified by the designers of the font. The concept of internal leading is specific to Windows.

The following figure shows internal and external leading and their relationship to the height of a font. The names beginning with the letter "tm" are members of the TEXTMETRIC structure.

## Character Sets

All fonts use a character set. A character set contains punctuation marks, numerals, uppercase and lowercase letters, and all other printable characters. Each element of a character set is identified by a number.

Most character sets used in Windows are supersets of the U.S. ASCII character set, which defines characters for the 96 numeric values from 32 through 127. There are four major groups of character sets:

- Windows
- OEM
- Symbol
- Vendor-specific

### Windows Character Set

The Windows character set is the most commonly used character set in Windows programming. It is essentially equivalent to the ANSI character set. The blank character is the first character in the Windows character set. It has a hexadecimal value of 0x20 (decimal 32). The last character in the Windows character set has a hexadecimal value of 0xFF (decimal 255).

Many fonts specify a default character. Whenever a request is made for a character that is not in the font, GDI provides this default character. Many fonts using the Windows character set specify the period (.) as the default character. TrueType fonts typically use an open box as the default character.

Fonts use a break character to separate words and justify text. Most fonts using the Windows character set specify the blank character, whose hexadecimal value is 0x20 (decimal 32).

For Windows version 3.1, 24 characters have been added to the Windows code page:

Character	Name	Windows character code
,	base line single quote	130
<i>f</i>	florin	131
„	base line double quote	132
...	ellipsis	133
†	dagger	134
‡	double dagger	135
^	circumflex	136
‰	per mille	137
Š	S Hacek	138
<	left single guillemet	139
Œ	OE ligature	140
‘	left single quote	145
’	right single quote	146
“	left double quote	147
”	right double quote	148
•	bullet	149
–	en dash	150
—	em dash	151
~	tilde	152
™	trademark ligature	153
š	s Hacek	154
>	right single guillemet	155
œ	oe ligature	156
ÿ	Y Dieresis	159

The characters for left and right single quote were added to the character set for the release of Windows version 3.0.

### OEM Character Set

The OEM character set is typically used in full-screen MS-DOS sessions for screen display. Characters 32 through 127 are usually the same in the OEM, U.S. ASCII, and Windows character sets. The other

characters in the OEM character set (0 through 31 and 128 through 255) correspond to the characters that can be displayed in a full-screen MS-DOS session. These characters are generally different from the Windows characters.

#### Symbol Character Set

The Symbol character set contains special characters typically used to represent mathematical and scientific formulas.

#### Vendor-Specific Character Sets

Many printers and other output devices provide fonts based on character sets that differ from the Windows and OEM sets--for example, the EBCDIC character set. To use one of these character sets, the printer driver translates from the Windows character set to the vendor-specific character set.

## Fonts in Windows

Windows applications can use three different kinds of font technologies to display and print text. This section discusses these font technologies and gives Windows-specific background information about fonts.

### Raster, Vector, and TrueType Fonts

Previous versions of Windows had two types of fonts: raster and vector. Windows version 3.1 introduces a third type--TrueType fonts.

Raster fonts are stored as bitmaps. These bitmaps are designed for output devices of a particular resolution. GDI typically synthesizes bold, italic, underline, and strikeout characteristics for raster fonts; however, the results are not always attractive. When GDI must change the size of a raster font, aliasing problems can also reduce the attractiveness of the text. Raster fonts are useful for specialized applications in which TrueType fonts are not available. Another possible advantage to using raster fonts derives from the large number of raster fonts that are often present on a user's system; an application could look for the name of a particular specialized or decorative font and use a TrueType font if the specified font was not present.

When an application requests an italic or bold font that is not available, GDI synthesizes the font by transforming the character bitmaps. When an application using only raster fonts requests a point size that is not available, GDI also transforms the bitmaps to produce the font. Because TrueType font families include bold, italic, and bold italic fonts, and because TrueType fonts are scalable to any requested point size, GDI does not synthesize fonts as frequently as it did for earlier versions of Windows.

Windows version 3.1 contains a new set of raster fonts. This set, called Small Fonts, is for use at resolutions of less than 8 points. Although TrueType fonts can be scaled to less than 8 points, glyphs this small may not be legible enough for regular use. Because glyphs this small contain very little detail, it is more efficient to use the raster small fonts than to scale TrueType fonts to the small size. (GDI synthesizes bold and italic attributes for the raster small fonts, when necessary.)

Vector fonts are stored as collections of GDI calls. They are time-consuming to generate but are useful for such devices as plotters, on which bitmapped characters cannot be used. (By drawing lines, GDI can simulate vector fonts on a device that does not directly support them.) Prior to the introduction of TrueType fonts, vector fonts were also useful for applications that used very large or distorted characters or characters that needed to be perpendicular to a base line that was at an angle across the display surface.

TrueType fonts are stored as collections of points and hints that define character outlines. (Hints are algorithms that distort scaled font outlines to improve the appearance of the bitmaps at specific resolutions.) When an application requests a TrueType font, the TrueType rasterizer uses the outline and the hints to produce a bitmap of the size requested by the application.

The default font for a device context is the System font, a proportionally spaced raster font representing characters in the Windows character set. Its font name is System. Windows uses the System font for menus, window titles, and other text.

It is possible to have multiple fonts in the system that have the same name (for example, a Courier device font and a Courier GDI raster font). However, applications typically do not present a font name to the user more than once--instead, they discard duplicates. Applications can control which font is presented to the user when duplicate font names occur by using the `lfOutPrecision` member of the `LOGFONT` structure.

### Font Resource Files

The SYSTEM subdirectory of a user's Windows directory (the directory in which Windows is installed) contains the system's font resource files. A font resource file is an empty Windows library; it contains no code or data but does contain resources.

Raster and vector font resource files are identified by the .FON filename extension. TrueType font resource files have the .FOT filename extension. Each .FOT file is a relatively short header that refers to a file containing TrueType font information. These TrueType font-information files have the same base filename as the .FOT files, but have the .TTF filename extension.

Some of the filenames for raster and vector fonts are followed by a lowercase letter that indicates the resolution for which the font was designed. This letter varies according to the type of display device that was specified when the fonts were installed. Following are the lowercase letters used to identify different resolutions:

Letter	Device
a	CGA

b	EGA
c	Okidata printers
d	IBM and Epson printers
e	VGA
f	IBM 8514/A

For more information about the format of font resource files, see [Resource Formats](#).

### Basics of TrueType Fonts

The TrueType fonts incorporated into Windows 3.1 are much more versatile than the fonts that were available in previous versions of Windows. TrueType fonts can be scaled and rotated; they allow the same fonts to be used on the screen as are used on printers; and they allow documents to be portable between printers, applications, and systems.

The following table lists the 13 core TrueType fonts distributed with Windows version 3.1. (Windows 3.1 may include additional TrueType fonts that supplement this core set.)

Font family	Font name	Type
Arial	Arial	Sans serif, variable pitch
	Arial Bold	Sans serif, variable pitch
	Arial Italic	Sans serif, variable pitch
	Arial Bold Italic	Sans serif, variable pitch
Courier New	Courier New	Serif, fixed pitch
	Courier New Bold	Serif, fixed pitch
	Courier New Italic	Serif, fixed pitch
	Courier New Bold Italic	Serif, fixed pitch
Symbol®	Symbol	N/A
Times New Roman	Times New Roman	Serif, variable pitch
	Times New Roman Bold	Serif, variable pitch
	Times New Roman Italic	Serif, variable pitch
	Times New Roman Bold Italic	Serif, variable pitch

TrueType font technology offers many benefits to application designers, at little or no cost. It is not necessary to revise an application written for Windows version 3.0 for that application to use TrueType fonts. If you want your application to take full advantage of the greater precision and versatility available with TrueType fonts, however, you can use the following new font functions:

Function	Description
<a href="#">CreateScalableFontResource</a>	Creates a font resource file for a specified TrueType font.
<a href="#">EnumFontFamilies</a>	Retrieves the fonts available on a specified device.
<a href="#">GetCharABCWidths</a>	Retrieves the widths of consecutive TrueType characters.
<a href="#">GetFontData</a>	Retrieves font-metric data (or the entire font) from a TrueType font file.
<a href="#">GetGlyphOutline</a>	Retrieves data describing an individual character in a TrueType font.
<a href="#">GetOutlineTextMetrics</a>	Retrieves font metrics for TrueType fonts.
<a href="#">GetRasterizerCaps</a>	Determines whether TrueType is installed.

### Benefits of TrueType

TrueType fonts offer many advantages over previous font technologies for Windows:

- What you see is what you get (WYSIWYG).
  - Applications can scale and rotate TrueType fonts. TrueType fonts are attractive at all sizes. An application can use the same fonts on the screen and the printer.
- Printer portability.
  - TrueType fonts work on different printers. Because detailed font metrics are available, an application can compose documents in a device-independent fashion.
- Document portability.
  - Applications can embed TrueType fonts in documents. TrueType fonts work on different platforms. Applications can use the detailed font metrics to compose documents in a platform-independent fashion.

- **Simplicity.**

The versatility of TrueType fonts reduces the number of required choices and compromises.

TrueType solves two important problems: matching fonts to the printer in use, and presenting high-quality fonts at any size on all devices.

The most obvious benefit of TrueType fonts is that they are scalable. Users can use TrueType to get virtually any point size they like. With TrueType, Windows users no longer need to think about the availability of point sizes on their printer or screen, about running a utility to create raster fonts, or about disk storage for these bitmaps.

TrueType fonts are presented to applications through the same enumeration and selection functions as the raster fonts. As a result, TrueType fonts work with every Windows application. Windows printer drivers have also been modified as required to support the use of TrueType.

#### Compatibility with Earlier Windows Versions

The introduction of TrueType fonts introduces a few issues that are important for applications developed for earlier versions of Windows.

#### Identifying TrueType Fonts for Users

Before TrueType fonts were introduced, some users had many different fonts to choose between; now, these users have still more choices. (Users can simplify their choices by selecting the "Enable TrueType Fonts" and "Show Only TrueType Fonts in Applications" check boxes in the Fonts dialog box from Control Panel.) Applications can use the standard font dialog box to make it easier for users to manage the fonts on their systems.

#### Character Widths

TrueType fonts use ABC character spacing, a spacing method that does not rely on the width of a character cell and any overhang (the method used for raster fonts). The extra accuracy of ABC spacing can introduce a problem for applications written prior to Windows version 3.1. Older applications that use character widths instead of ABC widths with TrueType fonts incorrectly calculate the end of the last glyph in the line. This calculation could be off by as much as several pixels. It is also possible that a line could start slightly to the left of the starting point specified by the application. These inaccuracies sometimes lead to problems when the screen is redrawn or when a selection of text is highlighted; pieces of glyphs can be handled incorrectly at either end of a line of text.

Many applications written before TrueType became available use the ExtTextOut function to clip or redraw lines of text that extend beyond the visible margins of the document. This method prevents any extra pieces of glyphs from being left behind because of incorrect character-width calculations.

#### MS Serif and MS Sans Serif Fonts

In Windows version 3.1, the raster fonts Tms Rmn and Helv have been replaced by identical fonts named MS Serif and MS Sans Serif, respectively. The Tms Rmn and Helv font names are mapped to their replacements in a new section of WIN.INI called [FontSubstitutes]. Whenever an application requests Helv or Tms Rmn, the font mapper checks this section and makes the appropriate substitution. The [FontSubstitutes] section also maps Helvetica@ to Arial and Times@ to Times New Roman.

A user could change the [FontSubstitutes] section to map any font name to any other font name. For example, a user could map Tms Rmn and Helv to the Times New Roman and Arial TrueType fonts. Entries in [FontSubstitutes] do not change the names of fonts, however; a user could not force Arial to appear as Helvetica in font menus.

The EnumFonts and EnumFontFamilies functions use the [FontSubstitutes] section of WIN.INI so that applications written prior to Windows version 3.1 do not fail unexpectedly when enumerating preexisting font names. If an application specifies Helv in a call to EnumFontFamilies, GDI enumerates the available MS Sans Serif fonts. When an application calls either of these functions with a NULL family name, GDI enumerates a representative font from each available family, returning the actual names of the fonts, not the remapped names.

Because most Windows applications display font menus that include only the fonts that can be printed on the current printer, this change in font names does not affect most users. Only users of dot-matrix printers see the new names in font menus and dialog boxes.

#### Font-Height Metrics Can Depend on Attributes

Because the members of a TrueType font family, such as bold and italic, come from different outlines, in some cases the font-height metrics could be different within a TrueType font family. For raster fonts this is not a problem, because when Windows simulates attributes, these metrics are preserved, and

because hand-tuned bitmaps were made with matching heights. For the set of fonts shipped with Windows 3.1, most (but not all) of the height metrics match.

### Text and Character Attributes

Character attributes are such features as whether a character is bold or italic and whether it has serifs. Text attributes are such features as line and character spacing and text justification. This section introduces some of these attribute categories. For descriptions of individual attributes, see the descriptions of the LOGFONT, NEWTEXTMETRIC, TEXTMETRIC, and OUTLINETEXTMETRIC structures in the Microsoft Windows Programmer's Reference, Volume 3.

### Character Width

#### Line and Character Spacing

Before the introduction of TrueType fonts, it was difficult for an application to position characters exactly, especially if the characters were in a string that included bold or italic text. Instead of the width of the character glyph, most Windows functions use the advance width of characters, which includes space on either side of the glyph, as in the following figure:



Applications can control the spacing of TrueType characters accurately by using ABC character spacing. GDI constructs ABC spacing from information provided by the TrueType rasterizer. The "A" spacing is the width to add to the current position before placing the glyph. The "B" spacing is the width of the glyph itself. The "C" spacing is the white space to the right of the glyph. The total advance width is given by A+B+C.

Because either or both of the A and C increments can be negative, characters can overhang or underhang the character cell in a way that was not previously possible with GDI. For example, in the following illustration, A and C increments for the letter "f" are all negative, but the A and C increments for the letter "g" are positive.

When an application using TrueType fonts calls a text-output function, GDI uses the font's complete set of ABC widths to provide character-placement information to the device driver.

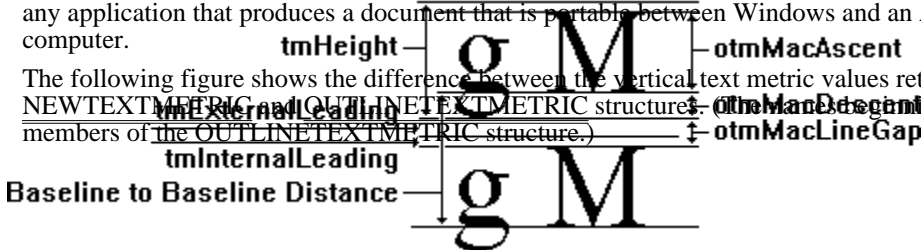
Some applications determine the line spacing between text lines of different sizes by using a font's maximum ascender and descender. An application can retrieve these values by calling the GetTextMetrics function and then checking the tmAscent and tmDescent members of the NEWTEXTMETRIC structure.

The maximum ascender and descender are different from the typographic ascender and descender; in TrueType fonts, the typographic ascender and descender are typically the top of the "f" glyph and bottom of the "g" glyph. Rounded characters typically extend slightly beyond the limits of characters with straight edges, to overcome an optical illusion that would make them appear too small otherwise. An application can retrieve the typographic ascender and descender for a TrueType font by calling the GetOutlineTextMetrics function and checking the values in the otmAscent and otmDescent members of the OUTLINETEXTMETRIC structure.

Applications that use the HPPCL5A printer driver may experience problems with line spacing for the scalable fonts that are built into the HP LaserJet III printer. These fonts use external leading in the place of internal leading; accent marks for capital letters print outside the character cell reported by the tmHeight member of the NEWTEXTMETRIC structure.

TrueType font metrics do not correspond exactly to the metrics for Windows raster fonts, because TrueType font metrics have been designed by Apple Computer, Inc. TrueType metrics are required for any application that produces a document that is portable between Windows and an Apple Macintosh computer.

The following figure shows the difference between the vertical text metric values returned in the NEWTEXTMETRIC and OUTLINETEXTMETRIC structures. (The values with "otm" are members of the OUTLINETEXTMETRIC structure.)



The overhang added by GDI when it synthesizes a bold or italic font is not taken into account by the GetTextExtent function.

#### Logical and Physical Inches

A logical inch is a measure Windows uses for presenting legible fonts on the screen; it is generally 30 to 40 percent larger than a physical inch. A 10-point font on a screen is larger than a 10-point font produced by a printer. Fonts on the screen are made larger because most screens do not have high enough resolutions to make a 10-point font legible. Furthermore, users generally read text on screens from a greater distance than they read text on paper.

Although logical inches solve the problem of legible fonts on the screen, they prevent a perfect match between the output of the screen and printer. The text on a screen is not simply a scaled version of the text that will appear on the page, particularly if graphics are incorporated into the text.

An application can retrieve the physical dimensions of a font by calling the GetOutlineTextMetrics function. To determine the dimensions of an output device, an application can call the GetDeviceCaps function. GetDeviceCaps returns both physical and logical dimensions.

#### Font Sizes

Most Windows applications use the MM\_TEXT mapping mode instead of MM\_TWIPS, because MM\_TEXT makes possible a relatively simple conversion from logical to physical font sizes. With MM\_TEXT, each logical unit is mapped to one pixel.

To determine the point size for a font, an application must first convert the information returned in the NEWTEXTMETRIC or OUTLINETEXTMETRIC structure using the size of the logical inch for the output device. For example, an application using MM\_TEXT units might use a font that has a cell height (tmHeight) of 12 and an internal leading (tmInternalLeading) of 2. The cell height minus the internal leading gives the point size in logical units; in this case, the point size of the font is 10 units (pixels).

To convert this value into a typographic point size (that is, a value in which one point equals 1/72 inch), the application should use the GetDeviceCaps function to determine the vertical size and resolution of the screen and the number of pixels per logical inch supported by that device. For example, if an application working in MM\_TEXT mapping mode requires a 12-point font, it could use the value produced by the following algorithm in the lfHeight member of the LOGFONT structure:

$$-1 * ((LOGPIXELSY * 12) / 72)$$

Using a negative value in the lfHeight member causes GDI to use the value as the height of the character glyphs, not the height of the character cell. The LOGPIXELSY value is returned by a call to the GetDeviceCaps function. The point size of the requested font is 12, and the number of points in a physical inch is 72.

Similarly, an application could use the following algorithm to determine the point size of a font from information returned in the NEWTEXTMETRIC structure:

$$((tmHeight - tmInternalLeading) * 72) / LOGPIXELSY$$

#### Font Mapper

When calling a font-creation function, an application describes the font either by using a LOGFONT structure in a call to the CreateFontIndirect function or by using the parameters of the CreateFont function. The font returned by these functions is called a logical font, because a font matching the described characteristics is not necessarily available in the system. GDI uses the logical font to create a physical font, by finding the closest match to the logical font among the available TrueType, raster, vector, and device-dependent fonts.

The Windows font mapper determines which of the available fonts is the closest match to the requested logical font. The font mapper often chooses a TrueType font as the closest match; it will choose a raster or vector font only when the logical font matches the characteristics of the raster or vector font very closely or when the logical font specifies the name of the raster or vector font. Typically, a TrueType font is chosen when it is specifically requested or when GDI would otherwise have to synthesize the font. For example, if a font name is not specified in the logical font or if the specified name does not exist, the font mapper chooses a TrueType font that matches the requested point size, serif characteristics, and pitch.

When the font mapper determines that a TrueType font is the closest match for a requested logical font, the TrueType engine produces enhanced GDI raster characters that are presented to the raster device. (The characters are enhanced by the use of ABC character widths.) For devices that do not have raster



font capabilities, the driver must request the TrueType engine to provide the glyphs in a form the driver can use.

When the font mapper chooses between raster fonts, it chooses the font that is closest to the requested size without being larger than that size.

When an application requests a very small font, the font mapper may choose one of the small fonts stored in the SMALLX.FON font resource file. TrueType fonts specify a suggested minimum size, which can be retrieved by calling the `GetOutlineTextMetrics` function and checking the `otmusMinimumPPEM` member of the `OUTLINETEXTMETRIC` structure. When an application requests a font smaller than this size, the font mapper typically chooses a small font instead of a TrueType font. If the requested size is not available as a small font, however, GDI scales the TrueType font instead. Microsoft's 13 core TrueType fonts are designed to be readable as small as 8 points on a VGA screen, although they can be used at smaller sizes.

### Standard Font Dialog Box

Windows applications should take advantage of the standard font dialog box for Windows 3.1.

Following are the advantages of this dialog box:

- It shows the user the font family name (for example, Times New Roman) along with the styles (for example, Regular, Bold Italic, and other combinations of italic and weight) for the installed fonts.
- It allows Windows version 3.0 simulations and effects to be applied, if the user wants them. When bold or italic simulations are applied, the user is warned that the font may not print as selected.
- It displays weights or styles outside the four standard styles (regular, bold, italic, bold italic).
- It clearly tells the user which fonts are TrueType and which are not.

The standard font dialog box also introduces a consistent user interface and frees applications from having to implement their own dialog boxes for fonts, while retaining enough flexibility for applications to add custom controls. The dialog box looks like this:

-

## TrueType Font Technology

With TrueType fonts, applications have much greater control over the final appearance of documents than was possible with previous Windows font technologies. Much of this added control is the result of the portability of TrueType fonts: An application can move them from the system to a printer, from a printer to the system, from one system to another system (by "embedding" them in documents), and even port them between incompatible operating systems.

Sophisticated desktop-publishing and word-processing applications go to great lengths to make the screen output mimic the printer output. Some applications even change the way the fonts appear on the screen, in an attempt to show users what the printer output will look like. This kind of application benefits greatly from exploiting the advantages of TrueType.

### What You See Is What You Get: WYSIWYG

WYSIWYG means that the screen output matches the printer output. With perfect WYSIWYG, the user would be able to place a page of printed output over the same screen output and see every character and graphic element in exactly the same place. If the screen and printer have different resolutions, however, this degree of matching is impossible. Usually, WYSIWYG simply means that line breaks, paragraph breaks, and page breaks are the same on both devices and that justified paragraphs are presented properly. WYSIWYG does not mean the same document on two different printers will be formatted in exactly the same way. Because most applications make the best use of the available printer, WYSIWYG often applies only to the correspondence between the screen and printer for a given printer.

TrueType offers a higher level of WYSIWYG than was available with earlier versions of Windows, because it works on every device. Most Windows applications lay out the screen based on the target printer. The fonts they enumerate for the user are the fonts that can be printed. Because TrueType fonts work on the target printer, they are enumerated by the printer driver to the application and are typically displayed to the user as printer fonts. When the application and GDI match screen fonts to the printer fonts, the TrueType fonts are used on the screen as well.

If no screen font matches the widths of characters in the chosen printer font, WYSIWYG is difficult to achieve. When this happens, applications sometimes make the average width of the characters match, with as little variation in specific characters as possible. More exact matching is achieved with a technique known as metric coercion. There are two basic methods of coercing character metrics: width coercion and shape coercion. Width coercion simply adjusts the spacing between words and characters, and shape coercion applies a transformation to each character to force it into a bounding box. Because shape coercion can lead to unacceptably deformed characters, width coercion is typographically preferred.

Although Windows does not include a function to deform individual characters, the `lfWidth` member of the `LOGFONT` structure allows an application to scale the width of a TrueType font independently of its height. (Most applications do not scale TrueType fonts in this manner, however, because the results are usually unattractive.)

### Embedded Fonts

Embedding a font is the technique of bundling the fonts used by a document into the document itself for transmission to another computer. Embedding a font guarantees that a font specified in a transmitted document will be present on the computer receiving the document. Not all fonts can be moved from computer to computer, however, since most fonts are licensed to only one computer at a time. In Windows, only TrueType fonts can be embedded.

Applications should embed a font in a document only on request from a user. An application cannot be distributed along with documents that contain embedded fonts, nor can an application itself contain an embedded font. Whenever an application distributes a font, in any format, the proprietary rights of the owner of the font must be acknowledged.

A font's license may not allow embedding; it may give read-write permission for a font to be installed and used on the destination computer; or it may give read-only permission. Read-only permission allows a document to be viewed and printed (but not modified) by the destination computer; documents with read-only embedded fonts are themselves read-only. Read-only embedded fonts may not be unbundled from the document and installed on the destination computer.

Applications that support embedded fonts determine the license status of a font by checking the `otmfsType` member of the `OUTLINETEXTMETRIC` structure. If bit 1 of `otmfsType` is set, embedding is not permitted for the font. If bit 1 is clear, the font can be embedded. If bit 2 is set, the embedding is read-only.

It may be a violation of a font vendor's proprietary rights and/or user license agreement to embed any fonts for which embedding is not permitted or to fail to observe the following guidelines on embedding fonts.

#### Embedding a Font in a Document

When an application has determined that a font can be embedded, it can use the GetFontData function to read the font file. (Setting the dwTable and dwOffset parameters of GetFontData to 0L and the cbData parameter to -1L ensures that the application will read the entire font file, starting at the beginning of the font).

After retrieving the font data, the application can store it with the document, using any applicable format. Most applications build a font directory in the document, listing which fonts are embedded and whether the embedding is read-write or read-only. (An application can use the otmpStyleName and otmFamilyName members of the OUTLINETEXTMETRIC structure to identify the font.)

If the read-only bit is set for the embedded font, applications must encrypt the font data before storing it with the document. The encryption method need not be complicated; for example, using the XOR operator to combine the font data with an application-specified constant is adequate and fast.

#### Installing and Using an Embedded Font

An embedded font must be separated from the containing document and installed in the user's system before Windows can use it. Although the exact procedure for separating the font from the document depends on the method the application uses to embed it, the following three steps are always taken:

- 1 Resolve name conflicts before installing the font.
- 2 Write the font data to a file, decoding read-only fonts as necessary.
- 3 Use the CreateScalableFontResource function to create a font resource file for the unembedded font.

An application should avoid installing a font with the same name as a preexisting font. To determine whether there is duplication in style names, an application could compare the information returned by EnumFontFamilies against the family name and style name stored with the embedded font.

Embedded fonts that have read-write permission (that is, that can be permanently installed on the user's system) should be written to a file that has the .TTF filename extension. Embedded fonts with read-only permission should not use the .TTF extension and should avoid the .FOT and .FON extensions. (A typical filename extension for read-only embedded fonts is .TTR.) Because files for read-only embedded fonts must be removed from the system and from storage as soon as the containing document is closed, their names do not need to be meaningful except to the application.

Most applications put the files for embedded fonts that have read-write permission into either the SYSTEM subdirectory of the user's Windows directory or into the application's working directory. Files for read-only embedded fonts are typically put into a temporary directory.

Before installing an embedded font, an application must use the CreateScalableFontResource function to create a font resource file. Font resource files for fonts with read-write permission should use the .FOT filename extension. Font resource files for read-only fonts should use a different extension (for example, .FOR) and should be hidden from other applications in the system by specifying 1 for the first parameter of CreateScalableFontResource. The font resource files can be installed by using the AddFontResource function.

Applications should offer users the option of permanently installing embedded fonts that have read-write permission. To permanently install a font, applications should concatenate the family and style names and then use the WriteProfileString function to insert this string along with the .FOT file name in the [Fonts] section of the WIN.INI file. A typical font entry in the [Fonts] section looks like this example:

Times New Roman Bold (TrueType)=TIMESBD.FOT

If a document contains one or more read-only embedded fonts, the user must not be permitted to edit the document. If the user is allowed to edit the document in any way, the application must first strip away and delete the read-only embedded fonts. As mentioned earlier, read-only embedded fonts must be removed from the system and storage immediately when the document in which they were bundled is closed.

To delete read-only embedded fonts, an application should follow these steps:

- 1 Call the RemoveFontResource function for each font to be deleted.
- 2 Delete the font resource file for each font.
- 3 Delete each TrueType font file for each font.

When an application creating a file for a read-only embedded font specifies 1 for the first parameter of the `CreateScalableFontResource` function, the `EnumFonts` and `EnumFontFamilies` functions will not enumerate this font. Hiding read-only embedded fonts in this manner makes it unlikely that another application could use them, even though Windows resources are theoretically available to all processes in Windows. If an application does use a read-only embedded font installed by another application, it could be difficult for the installing application to delete the font. The `RemoveFontResource` function will not delete a font that is currently in use. In this case, an application should delete the resource file and the TrueType font file when the user closes the document that contained the read-only fonts.

It is very important that applications delete the TrueType font file for read-only embedded fonts. If the delete operation fails when the user closes the document, the application should periodically attempt to delete the file as the application runs, when it closes, and the next time it starts.

In some cases, an application could be unable to delete a TrueType font file for a read-only embedded font because of external events (such as a system failure). There is no legal liability for events that are out of the control of the application.

#### Printer Portability

A document with printer portability is formatted identically on all output devices under Windows--all monitors and all printers. Although TrueType allows the same font to be used on all output devices, this does not guarantee that line breaks will be the same on all devices. For line breaks to match, applications must take advantage of TrueType design metrics. These design metrics allow an application to compute the fractional portion of the spacing at the ends of lines and make up the difference in the interword spacing. This computation reduces the round-off error from a half-pixel per character to a half-pixel per line, preserving line breaks in all cases.

#### Line Breaks and Justification

Applications must cooperate in order to guarantee the printer portability enabled by TrueType technology, because different devices may have different resolutions. Even when fonts are portable across printers, glyphs designed or rasterized for different resolutions must have different pixel widths. For applications that use the `TextOut` function, for example, different character widths can lead to accumulated round-off errors that change line breaks and paragraph placement.

Applications that lay out a document at the highest printer resolution attempt to distribute any difference in character resolutions in white spaces. This method is not always successful; for example, it fails when all glyphs are one pixel larger at 600 dots per inch (DPI) than at 300 DPI. In this case, fonts with a width of 45 at 600 DPI would have a width of 23 at 300 DPI, a width of 11 at 150 DPI, and so on. There could easily be insufficient white space to absorb the glyphs at the lower resolutions if line breaks were being preserved, because the glyphs become larger in relation to the resolution of the device. In this case, the characters would have to overlap to preserve the line breaks. Even if all the character widths exactly doubled when changing from a resolution of 300 DPI to 600 DPI, the line breaks might not be the same if an application justified text--that is, aligned it on both the left and right. It is possible that another half-pixel of white space at the lower resolution would allow one more word on the line. At the higher resolution, the half-pixel would become a full pixel and the line breaks would change. (Similar device-resolution problems occur in the vertical direction.) TrueType exposes the design width of characters to help applications maintain line breaks.

Different printers, or even different production runs of the same printer, can have different limits for their printable areas. If a document has been laid out up to the margins of one printer, it may not format identically on a different printer. If glyphs are in contact with the margins on the first printer, parts of the glyph may be beyond the printable area on the second printer. Depending on the printer, the glyph will either be clipped or dropped completely.

Prior to the introduction of TrueType, sophisticated desktop-publishing and word-processing applications were forced to "reflow" the entire document whenever a user selected a different printer. Applications can now use TrueType font metrics to solve this problem.

#### Performance and Printer Portability

Printer portability can potentially downgrade font performance, quality, or both, depending on such factors as the type of connection between the computer and printer, the speed of the computer, the memory in the printer and the computer, the number of fonts being used, differing resolutions between the screen and printer, and the number of characters used in each font. Documents that are fully portable between printers necessarily cannot take advantage of the specialized features of a particular printer.

GDI cannot perform text operations to printer-compatible memory device contexts. This means that it is not possible to build a bitmap describing a page to be printed and then send the completed bitmap to the printer.

## Document Portability

A portable document appears the same on different operating systems. In the case of TrueType, documents can be portable between Windows and the Apple Macintosh computer; this could also be called platform portability. If a document appears the same on the Macintosh and with Windows, it can also look the same imported into different applications on either platform.

Since the same TrueType fonts work on the Macintosh, in Windows, and on all devices supported by both systems, the same characters and metrics could be exposed for all applications. Currently, however, fully portable documents are not possible. Windows and the Macintosh computer have slightly different character sets. Even though TrueType fonts contain the default Macintosh and PostScript character sets, Windows does not give applications access to the Macintosh characters. Likewise, a Macintosh application cannot gain access to the Windows characters present in TrueType fonts. Document portability is also a problem with international document exchange. Localized versions of TrueType fonts will still be in use for both the Apple System 7 and Windows version 3.1, leading to further character-set incompatibilities when documents that use these fonts are transmitted to a system that does not have them.

## Disk Space, Memory Usage, and Speed

An application's overall font performance could decrease if a large font cache forced the paging of more segments to the disk. With previous font technologies, this could occur even in situations that were not "low memory." Because fonts are cached glyph by glyph as they are used, however, less memory is used for the cache than would be required to keep the corresponding raster fonts in memory; this leads to a net performance gain. The only time the font cache uses more memory than fonts required in earlier versions of Windows is when multiple logical fonts would have been mapped to the same raster font. Typically, however, any additional swapping to disk caused by these larger caches is still faster overall than discarding and subsequently re-rendering bitmaps.

Hard-disk space is not a large problem for TrueType fonts, although more disk space is required for fonts with the introduction of TrueType. The two reasons for this increased space requirement are that raster fonts are shipped with TrueType fonts, for backward-compatibility reasons, and that users may have preexisting soft fonts on their hard disks.

Hard-disk space is not the only limitation imposed on TrueType fonts. GDI imposes an internal limit to the number of TrueType fonts that can exist simultaneously on a system. The maximum number of physical fonts is 1170. (The maximum number of logical fonts that can exist simultaneously on a system is 253.)

## Font Design and Scaling

Raster fonts are designed to be attractive and readable at a particular aspect ratio. (The aspect ratio is the ratio of the width and height of a pixel.) The digitized aspect of a font is the ideal x-aspect and y-aspect of that font. Windows provides an aspect-ratio filter to select fonts designed for a particular aspect ratio from all of the available fonts. The `GetAspectRatioFilter` function retrieves the setting for the current aspect-ratio filter. An application can use the `SetMapperFlags` function to change the algorithm the font mapper uses when it maps physical fonts to logical fonts.

The aspect ratio of the screen is not as critical for scalable fonts as it is for raster fonts. The dimensions of the em square for a TrueType font are used when scaling the font to a specified point size. (An em square is a square whose width is approximately equal to the width of the uppercase M.) Because the height of the em square is given in pixels, it can be thought of as the point size in device units. For example, a font could be referred to as a 50-ppem (pixels per em square) font. The pixel size determines the physical point size. For example, a 75-ppem font on a 300-DPI device is an 18-point font, while on a 150-DPI device it would be a 36-point font. The number of pixels required for the desired point size is computed by using the resolution of the output device and the em square size, according to the following formula:

$$\text{ppem} = (\text{PointSize}/72) * \text{DeviceResolution}$$

According to this formula, a 12-point font on a 72-DPI screen is at 12 ppem, while on a 300-DPI device it is at 50 ppem.

TrueType fonts can be scaled linearly, nonlinearly, or optically, depending on their design. Linear scaling means that the character width is scaled and rounded to the appropriate ppem. Nonlinear scaling means that hinted character widths can be larger or smaller than the scaled widths. Optical scaling is a superset of nonlinear scaling; it includes the preservation of the color and contrast of a font across point sizes. Optical scaling can involve changing the proportions of the stroke widths to preserve their perceived width and color.

The TrueType fonts shipped with Windows 3.1 scale nonlinearly. Windows applications can also support linearly and optically scaled TrueType fonts.

### Designing Portable Fonts

Most application developers need not be concerned with font-portability issues. This discussion is included here with other portability issues for those developers who need to create fonts that are portable between systems. Microsoft currently publishes a TrueType Font Files Specification, which teaches font vendors how to create a single TrueType font that will work in Windows, on the Macintosh computer, and in TrueImage.

Microsoft uses the same byte ordering in TrueType font files as Apple uses in its font files, to help make the fonts portable between the systems. As a result, Windows fonts can be moved directly to the Macintosh computer, where they can quickly be converted into font suitcases for installation. (The format of TrueType font files precisely follows the format of the Apple "sfnt" resource. To convert an MS-DOS binary TrueType font into an sfnt resource requires editing the file information, setting Type to sfnt and Creator to bass. The sfnt resource can then be integrated into a standard Macintosh font suitcase. To move a font suitcase to Windows, an application need only extract the sfnt portion from the data fork and move the suitcase, unaltered, to Windows. After the suitcase has been moved to Windows, it can be installed by using Control Panel or the CreateScalableFontResource and AddFontResource functions.

If a Macintosh font is installed that does not contain the Windows "cmap" mapping table, the system maps text fonts (for example, Times or ITC Zapf Chancery®) from the Macintosh character set onto the Windows character set. Novelty fonts (like ITC Zapf Dingbats®), which have no formal character set, are not mapped; these fonts are taken along with the Macintosh character encodings. The decision whether to remap is based on a test that looks at the "post" table (which contains PostScript names). Whenever necessary, Windows compensates for missing metric tables based on other metric data in the font; anything that cannot be computed in a reasonable manner is given a default value.

The creation of portable fonts requires more than just the right characters and the right character-mapping tables. All the metrics needed by all systems must be included and must yield the same results. Matching metrics for the individual characters is not a problem; since the characters and their hints and metrics appear only once in the TrueType font, the same metrics are available across platforms. The more difficult problems in the creation of portable fonts have to do with line-spacing metrics, the determination of font styles, and making these factors match across systems.

The Apple System 7 core TrueType fonts ship with metrics designed to be compatible with the raster fonts in System 6. The "hdmx" table will be used to force widths onto TrueType fonts that match those for the bitmaps at bitmap sizes. The "name" table (and its ability to group fonts by separating the family and subfamily names) is not used. (The name used comes from the FOND Macintosh font resource.) Only the macStyle bits (from the "head" table) denoting regular, bold, italic, or bold italic are used.

Apple's line spacing recommendations are less robust than the line-spacing used by Microsoft. The following formula defines the default recommended line spacing for a Macintosh font:

line spacing = ascent - descent + leading

The values for ascent, descent and leading come directly from TrueType values:

Macintosh	TrueType
ascent	otmMacAscent
descent	otmMacDescent
leading	otmMacLineGap

For its TrueType fonts, Apple recommends that Ascender - Descender = unitsPerEm, and LineGap = 0. This recommendation is based on the definition of point size for Macintosh raster fonts. Macintosh documentation defines the point size of a font as being equal to the line spacing (ascent - descent + leading). Although this definition is compatible with previous Apple font metrics, it ties line spacing to the size of the em square. Because some fonts (for example, Palatino) have ascenders and descenders that extend beyond the em square, the line-spacing definition is inconsistent for these fonts.

Windows and the Macintosh have the same default line spacing for a font only if the following formula is true:

$$\text{otmMacLineGap} \geq (\text{tmAscent} + \text{tmDescent}) - (\text{otmMacAscent} - \text{otmMacDescent})$$

Microsoft TrueType fonts follow this formula to ensure that default line spacing is preserved between the

Macintosh and Windows. The core fonts and all fonts from vendors that follow the Microsoft specification will have the same character widths, the same default line spacing, and the same character forms.

Unless the Windows and Macintosh font heights are equal, a font with a line gap of zero will yield different default line spacings in Windows and on the Macintosh.

Despite some incompatibilities, TrueType and GDI accept Macintosh-only fonts. Metrics that are not present in Macintosh-only fonts are set to default values. Although these default values are imperfect, using them allows Macintosh-only fonts to work in Windows.

## Using Fonts in Applications

The remainder of this topic discusses the implementation of font functions in Windows applications.

### Using Stock Fonts

GDI offers a variety of stock fonts that an application can retrieve and use. For many applications, the stock fonts provide all the functionality required for basic text output. To use stock fonts, an application specifies the type of font in the `GetStockObject` function. `GetStockObject` creates a handle to a logical font. When the application selects that handle into a device context, the font mapper uses the logical font to create a physical font. The application can select and use this physical font for text output.

GDI offers the following stock fonts:

Font	Description
<code>ANSI_FIXED_FONT</code>	Specifies a fixed-pitch font based on the Windows character set. A Courier font is typically used.
<code>ANSI_VAR_FONT</code>	Specifies a variable-pitch font based on the Windows character set. MS Sans Serif is typically used.
<code>DEVICE_DEFAULT_FONT</code>	Specifies a font preferred by the given device. Because this font depends on how the GDI font mapper interprets font requests, the font may vary widely from device to device.
<code>OEM_FIXED_FONT</code>	Specifies a fixed-pitch font based on an OEM character set. OEM character sets vary from system to system. For IBM computers and compatibles, the OEM font is based on the IBM PC character set.
<code>SYSTEM_FONT</code>	Specifies the System font. This is a variable-pitch font based on the Windows character set, and is used by the system to display window titles, menu names, and text in dialog boxes. The System font is always available. Other fonts are available only if they have been installed.

The following example retrieves a handle of the Windows variable stock font, selects it into a device context, and then writes a string using that font:

```
HWND hwnd, hOldFont;  
  
hFont = GetStockObject(ANSI_VAR_FONT);  
if (hOldFont = SelectObject(hdc, hFont)) {  
    TextOut(hdc, 10, 50, "Sample ANSI_VAR_FONT text.", 26);  
    SelectObject(hdc, hOldFont);  
}
```

If no other stock fonts are available, `GetStockObject` returns a handle to the System font (`SYSTEM_FONT`).

Applications that use the `GetStockObject` function to retrieve the handle of a logical font should work in `MM_TEXT` units. The logical font identified by the handle returned by `GetStockObject` may specify a height that does not match the height of the requested logical font when the application works in mapping modes other than `MM_TEXT`.

### Enumerating Fonts

An application can discover which fonts are available for a given device by using the `EnumFonts` or `EnumFontFamilies` function. These functions send information about the available fonts to a callback function that the application supplies. The callback function receives information in `LOGFONT` and `NEWTEXTMETRIC` structures. (The `NEWTEXTMETRIC` structure contains information about a TrueType font. When the callback function receives information about a non-TrueType font, the information is contained in a `TEXTMETRIC` structure.) By using this information, an application can allow the user to choose among only those fonts that are available.

The `EnumFontFamilies` function is similar to the `EnumFonts` function but includes some extra functionality. New and upgrading applications should use `EnumFontFamilies` instead of `EnumFonts`. `EnumFontFamilies` allows an application to take advantage of the style name that is available with TrueType fonts.

In previous versions of Windows, the only style attributes were weight and italic; any other styles were specified in the family name for the font. If an application used the `EnumFonts` function to query the available Courier fonts, for example, `EnumFonts` might return information for Courier, Courier Bold,



Courier Bold Italic, and Courier Italic, but it would not return information about any other Courier fonts that might be installed, because any other Courier fonts would typically have a different family name.

TrueType fonts are organized around a family name (for example, Courier New) and style names (for example, italic, bold, and extra-bold). The EnumFontFamilies function enumerates all the styles associated with a given family name, not simply the bold and italic attributes; if the system included a TrueType font called Courier New Extra-Bold, EnumFontFamilies would list it with the other Courier New fonts. The capabilities of EnumFontFamilies are helpful for fonts with many or unusual styles and for fonts that cross international borders. (Because a style name often changes with the language spoken in a country, an application that depends on the EnumFonts function could enumerate fonts whose names would change from country to country, while EnumFontFamilies would continue to enumerate the font families correctly.)

If an application does not supply a typeface name, the EnumFonts and EnumFontFamilies functions supply information about one font in each available family. To enumerate all the fonts in a device context, an application can specify NULL for the typeface name, compile a list of the available typefaces, and then enumerate each font in each typeface.

The following example uses the EnumFontFamilies function to retrieve the number of available raster, vector, and TrueType fonts:

```

FONTENUMPROC lpEnumFamCallBack;
UINT uAlignPrev;
int aFontCount[] = { 0, 0, 0 };
char szCount[8];

lpEnumFamCallBack = (FONTENUMPROC) MakeProcInstance(
(FARPROC) EnumFamCallBack, hInstApp);
EnumFontFamilies(hdc, NULL, lpEnumFamCallBack,
(LPARAM) aFontCount);
FreeProcInstance((FARPROC) lpEnumFamCallBack);

uAlignPrev = SetTextAlign(hdc, TA_UPDATECP);

MoveTo(hdc, 10, 50);
TextOut(hdc, 0, 0, "Number of raster fonts: ", 24);
itoa(aFontCount[0], szCount, 10);
TextOut(hdc, 0, 0, szCount, strlen(szCount));

MoveTo(hdc, 10, 75);
TextOut(hdc, 0, 0, "Number of vector fonts: ", 24);
itoa(aFontCount[1], szCount, 10);
TextOut(hdc, 0, 0, szCount, strlen(szCount));

MoveTo(hdc, 10, 100);
TextOut(hdc, 0, 0, "Number of TrueType fonts: ", 26);
itoa(aFontCount[2], szCount, 10);
TextOut(hdc, 0, 0, szCount, strlen(szCount));

SetTextAlign(hdc, uAlignPrev);
.
.
.

BOOL FAR PASCAL EnumFamCallBack(lplf, lpntm, FontType, aFontCount)
LPLOGFONT lplf;
LPNEWTEXTMETRIC lpntm;
short FontType;
LPSTR aFontCount;
{
    int far * aiFontCount = (int far *) aFontCount;

    if (FontType & RASTER_FONTTYPE)
        aiFontCount[0]++;
    else if (FontType & TRUETYPE_FONTTYPE)
        aiFontCount[2]++;
}

```

```

    else
    aiFontCount[1]++;

    if (aiFontCount[0] || aiFontCount[1] || aiFontCount[2])
    return TRUE;
    else
    return FALSE;
}

```

This example uses two masks, `RASTER_FONTTYPE` and `TRUETYPE_FONTTYPE`, to determine the type of font being enumerated. If the `RASTER_FONTTYPE` bit is set, the font is a raster font. If the `TRUETYPE_FONTTYPE` bit is set, the font is a TrueType font. If neither bit is set, the font is a vector font. A third mask, `DEVICE_FONTTYPE`, is set when a device (for example, a laser printer) supports downloading TrueType fonts; it is zero if the device is a display adapter, dot-matrix printer, or other raster device. An application can also use the `DEVICE_FONTTYPE` mask to distinguish GDI-supplied raster fonts from device-supplied fonts. GDI can simulate bold, italic, underline, and strikeout attributes for GDI-supplied raster fonts, but not for device-supplied fonts.

An application can also check bit 1 and 2 in the `tmPitchandFamily` member of the `NEWTEXTMETRIC` structure to identify a TrueType font. If bit 1 is zero and bit 2 is 1, the font is a TrueType font.

Vector fonts are categorized as `OEM_CHARSET` instead of `ANSI_CHARSET`. Some applications identify vector fonts by using this information, checking the `tmCharSet` member of the `NEWTEXTMETRIC` structure. This categorization usually prevents the font mapper from choosing vector fonts unless they are specifically requested. (Most applications do not use vector fonts, because they are slow and generally unattractive, and because TrueType fonts offer many of the same scaling and rotation features that required the use of vector fonts in earlier versions of Windows.)

#### Checking a Device's Text Capabilities

Applications can use the `EnumFonts` and `EnumFontFamilies` functions to enumerate the fonts in a printer-compatible memory device context. An application can also use the `GetDeviceCaps` function to retrieve information about the text capabilities of a device. By calling the `GetDeviceCaps` function with the `NUMFONTS` index, an application can determine the minimum number of fonts supported by a printer. (An individual printer may support more fonts than specified in the return value from `GetDeviceCaps` with the `NUMFONTS` index.) By using the `TEXTCAPS` index, an application can identify many of the text capabilities of the specified device.

The following example uses the `GetDeviceCaps` function to determine whether a device supports text rotation:

```

int result;

result = GetDeviceCaps(hdc, TEXTCAPS);
if (result & TC_CR_90)
    TextOut(hdc, 10, 100, "Device can rotate text 90 degrees",
33);
if (result & TC_CR_ANY)
    TextOut(hdc, 10, 120, "Device can rotate text at any angle",
35);
else if ((result & TC_CR_90) == 0 && (result & TC_CR_ANY) == 0)
    TextOut(hdc, 10, 100, "Device cannot rotate text", 25);

```

#### Creating a Logical Font

A logical font is a list of font attributes, such as height, width, character set, and typeface. An application creates a logical font to describe the font that is best suited for a given task; the font mapper uses this logical font to choose the available physical font that best matches the specified characteristics.

An application can use either the `CreateFont` or the `CreateFontIndirect` function to create a logical font. Most applications use `CreateFontIndirect`, assigning values to a `LOGFONT` structure. These functions return a handle of a logical font, which can then be selected into a device context and used.

The following example is a function that takes a handle of a device context, the name of a font, and a nominal point size as input. It creates a logical font of the requested size and face name and selects that font into the specified device context.

```

BOOL FAR PASCAL CreateLogFont(hdc, pszFace, PointSize)
HDC hdc;

```

```

PSTR pszFace;
int PointSize;
{
    HFONT hfnt, hfntOld;
    PLOGFONT plf = (PLOGFONT) LocalAlloc(GPTR, sizeof(LOGFONT));

    if (GetMapMode(hdc) != MM_TEXT) {
TextOut(hdc, 100, -200, "Mapping mode must be MM_TEXT",
28);
return FALSE;
    }

    plf->lfHeight = -MulDiv(PointSize,
GetDeviceCaps(hdc, LOGPIXELSY), 72);
    lstrcpy(plf->lfFaceName, pszFace);

    hfnt = CreateFontIndirect(plf);

    hfntOld = SelectObject(hdc, hfnt);

    . /* Use font for text output. */
    .
    LocalFree((LOCALHANDLE) plf);
    SelectObject(hdc, hfntOld);
    if (DeleteObject(hfnt))
return TRUE;
    else
return FALSE;
}

```

Memory for the logical font in this example is allocated and initialized to zero (by using the LocalAlloc function with the GPTR constant); this means the logical font created by the CreateFontIndirect function uses default values for all members except lfHeight and lfFaceName. (Applications should always specify values for at least these two members.) For a description of all of the members of the LOGFONT structure, see the Microsoft Windows Programmer's Reference, Volume 3.

The function in this example uses the Windows MulDiv function to convert the specified point size into a different negative value and then assigns that value to the lfHeight member. This conversion is required because logical inches are larger than physical inches. The MulDiv function multiplies the requested point size by the result of dividing the number of pixels per logical inch by the number of points in a physical inch (72). A negative value is specified for lfHeight to indicate that the system should interpret this value as the height of the character glyphs in the font; when a positive value is specified, GDI interprets it as the height of a font's character cells, including internal leading.

An application would use a positive value for the lfHeight member to choose a font that fits within a specific height. For example, to display a page in "print preview" mode, an application would retrieve the height of the printer font from the tmHeight member of the NEWTEXTMETRIC structure, scale that height to the screen resolution, and use this value for the lfHeight member. The formula in this case would look like this:

$$\text{lfHeight} = \frac{\text{tmHeight} * \text{DPI of screen}}{\text{DPI of printer}}$$

The results of this calculation should always be rounded down to the nearest whole number.

When an application specifies the handle of a logical font in a call to the SelectObject function, the font mapper returns a handle of the physical font that is the best match for the requested attributes.

An application that requires a raster font can identify the available raster fonts by calling the EnumFontFamilies function and checking the RASTER\_FONTTYPE bit. The application can then specify the typeface name in a LOGFONT structure. Similarly, vector fonts can be selected by checking the RASTER\_FONTTYPE and TRUETYPE\_FONTTYPE bits. An application can also specify a vector font by specifying OEM\_CHARSET in the lfCharSet member of the LOGFONT structure, as discussed in Section 18.4.2, "Enumerating Fonts."

An application can use TrueType fonts exclusively by specifying `OUT_TT_ONLY_PRECIS` in the `lfOutPrecision` member of the `LOGFONT` structure. This is important for applications that use object linking and embedding (OLE), because metafiles can be scaled much better when they use only TrueType fonts.

#### Retrieving Information About the Selected Font

Applications can retrieve font information from a device context by using the `GetTextMetrics`, `GetTextFace`, and `GetOutlineTextMetrics` functions.

The `GetTextMetrics` function copies a `TEXTMETRIC` structure into a buffer. The `TEXTMETRIC` structure contains a description of the physical font, including the average dimensions of the character cells within the font, the spacing between lines of text, the number of characters in the font, and the character set on which the font is based. An application working with TrueType fonts can call the `GetOutlineTextMetrics` function to retrieve information in an `OUTLINETEXTMETRIC` structure.

Applications often use the `TEXTMETRIC` structure to determine how much space to specify between lines of text. For example, to compute an appropriate value for single-line spacing, an application could add the values of the `tmHeight` and `tmExternalLeading` members. The `tmHeight` member specifies the height of each character cell, and `tmExternalLeading` specifies the font designer's recommended spacing between the bottom of one character cell and the top of the next. (More accurate information can be retrieved for TrueType fonts from the `OUTLINETEXTMETRIC` structure; in this case, applications can add the values of the `otmAscent`, `otmDescent`, and `otmLineGap` members.) The following example writes several lines of single-spaced text:

```
TEXTMETRIC tm;
int LineSpacing, i, YIncrement;

GetTextMetrics(hdc, &tm);
LineSpacing = tm.tmHeight + tm.tmExternalLeading;

YIncrement = 50;
for (i = 0; i < 4; i++) {
    TextOut(hdc, 10, YIncrement, "Single-line spacing", 19);
    YIncrement += LineSpacing;
}
```

The `GetTextFace` function copies a name identifying the typeface of the selected font into a buffer. An application can use this information in dialog boxes and menus.

#### Retrieving Information About a Logical Font

An application can retrieve information about a font by specifying the font handle in a call to the `GetObject` function. The `GetObject` function copies logical-font information to a `LOGFONT` structure.

The following example uses the `GetObject` function to retrieve logical-font information for a font and then checks whether the font is italic:

```
LOGFONT lf;

GetObject(hfnt, sizeof(LOGFONT), &lf);
if (lf.lfItalic)
    return TRUE;
else
    return FALSE;
```

#### Drawing Text

An application can use the following functions to draw text:

Function	Description
<code>DrawText</code>	Draws formatted text in a rectangle. <code>DrawText</code> formats text by expanding tabs into appropriate spaces, aligning text to the left, right, or center of the given rectangle, and breaking text into lines that fit within the given rectangle. This is not a GDI function; it is in <code>USER.EXE</code> .
<code>ExtTextOut</code>	Writes a character string within a rectangular region. The rectangular region can be opaque (filled with the current background color), and it can be a clipping region.

<u>GrayString</u>	Draws gray text by writing the text in a memory bitmap, graying the bitmap, and then copying the bitmap to the device. <u>GrayString</u> grays the text regardless of the selected brush and background. This is <u>not</u> a GDI function; it is in USER.EXE.
<u>TabbedTextOut</u>	Writes a character string, expanding tabs to the values specified in an array of tab-stop positions.
<u>TextOut</u>	Writes a character string at a specified location.

The ExtTextOut function is the fastest Windows text-output function. The DrawText function is the slowest (although it offers the richest formatting options).

Instead of using the GrayString function, an application could simply set the text color to gray, as follows:

```
dwColorPrevious = SetTextColor(hdc, GetSysColor(COLOR_GRAYTEXT));
```

#### Setting the Text Alignment

An application can query and set the text alignment for a device context by using the GetTextAlign and SetTextAlign functions. The text-alignment settings determine how text is positioned relative to a given location. Text can be aligned to the right or left of the position or centered over it; it can also be aligned above or below the point. In addition, an application can use the SetTextAlign function to update the current position when a text-output function is called.

For example, the following example uses the SetTextAlign function to update the current position when the TextOut function is called. In this example, cArial is an integer that specifies the number of Arial fonts:

```
UINT uAlignPrev;
char szCount[8];

uAlignPrev = SetTextAlign(hdc, TA_UPDATECP);
MoveTo(hdc, 10, 50);
TextOut(hdc, 0, 0, "Number of Arial fonts: ", 23);
itoa(cArial, szCount, 10);
TextOut(hdc, 0, 0, (LPSTR) szCount, strlen(szCount));
SetTextAlign(hdc, uAlignPrev);
```

#### Using Color

When an application first creates a device context, the text color is black and the background color is white. An application can add color to text by setting the text and background colors of the device context. The text color determines the color of the character to be written; the background color determines the color of everything in the character cell except the character.

An application can set the text and background colors by using the SetTextColor and SetBkColor functions. The following example sets the text color to red and the background color to green:

```
SetTextColor(hdc, RGB(255,0,0));
SetBkColor(hdc, RGB(0,255,0));
```

The background color applies only when the background mode is opaque. The background mode determines whether the background color in the character cell has any effect on what is already on the screen. If the mode is opaque, the background color overwrites anything already on the screen; if the mode is transparent, anything on the screen that would otherwise be overwritten by the background is preserved. The background color for an italic string that GDI has synthesized is sheared along with the characters; this can lead to unexpected results when the text background color is different from the window background color. An application can set and retrieve the background mode by using the SetBkMode function and GetBkMode functions. Similarly, an application can retrieve the current text and background color by using the GetTextColor and GetBkColor functions.

#### Using Multiple Fonts in a Line

Different type styles within a font family can have different widths. For example, bold and italic styles of a family are always wider than the roman style for a given point size. An application that can display or print several type styles on a single line must keep track of the width of the line to avoid having characters print on top of one another.

An application can use the following functions to retrieve the width (or extent) of text in the current font:

Function	Description
----------	-------------

**GetTabbedTextExtent** Computes the width and height of a character string. If the string contains one or more tab characters, the width of the string is based upon a specified array of tab-stop positions.

**GetTextExtent** Computes the width and height of a line of text.

When necessary, GDI synthesizes a font by changing the character bitmaps. To synthesize a character in a bold font, GDI draws the character twice: once at the starting point, and again one pixel to the right of the starting point. To synthesize a character in an italic font, GDI draws the two rows of pixels at the bottom of the character cell, moves the starting point one pixel to the right, draws the next two rows, and continues until the character has been drawn. The base line of a synthesized italic character is shifted to the right by an amount determined by the height of the character cell. To determine the amount a base line is shifted to the right, an application can perform the following calculation, using values retrieved by a call to the **GetTextMetrics** function:

$$\text{units base line shifted right} = (\text{tmDescent} * \text{tmOverhang}) / \text{tmAscent}$$

One way to write a line of text that contains multiple fonts is to use the **GetTextExtent** function after each call to **TextOut** and add the length to a current position. The following example writes the line "This is a sample string.", using bold characters for the words "This is a", italic characters for the word "sample", and system default characters for "string.":

```
int XIncrement;
TEXTMETRIC tm;
HFONT hfntDefault, hfntItalic, hfntBold;

XIncrement = 10;
hfntDefault = SelectObject(hdc, hfntBold);
TextOut(hdc, XIncrement, 50, "This is a ", 10);

XIncrement += LOWORD(GetTextExtent(hdc, "This is a ", 10));
GetTextMetrics(hdc, &tm);
XIncrement -= tm.tmOverhang;
SelectObject(hdc, hfntItalic);
GetTextMetrics(hdc, &tm);
XIncrement -= tm.tmOverhang;
TextOut(hdc, XIncrement, 50, "sample ", 7);

XIncrement += LOWORD(GetTextExtent(hdc, "sample ", 7));
SelectObject(hdc, hfntDefault);
TextOut(hdc, XIncrement - tm.tmOverhang, 50, "string.", 7);
```

In this example, the **GetTextExtent** function returns a 32-bit value (of type **DWORD**) containing both the length and height of the specified string. The **LOWORD** macro then retrieves the length of the string, which is added to the current position. The **GetTextMetrics** function retrieves the overhang for the current font. Because the overhang is zero if the font is a TrueType font, the overhang value does not change the string placement in that case. For raster fonts, however, it is important to use the overhang value. The overhang is subtracted from the bold string once, to bring subsequent characters closer to the end of the string if the font is a raster font. Because overhang affects both the beginning and end of the italic string in a raster font, the glyphs begin to the right of the specified location and end to the left of the endpoint of the last character cell. (The **GetTextExtent** function retrieves the extent of the character cells, not the extent of the glyphs.) To account for the overhang for the raster italic string, this example subtracts the overhang before placing the string and subtracts it again before placing subsequent characters.

An application that must place characters with greater precision can use the **GetCharWidth** or **GetCharABCWidths** function to retrieve the widths of individual characters in a font. The **GetCharABCWidths** function is more accurate than the **GetCharWidth** function, but only when it is used with TrueType fonts; when **GetCharABCWidths** is used with non-TrueType fonts, it retrieves the same information as **GetCharWidth**.

The **SetTextJustification** function adds extra space to the break characters in a line of text. An application can use the **GetTextExtent** function to determine the extent of a string, subtract the extent from the total amount of space the line should occupy, and use the **SetTextJustification** function to distribute the extra space among the break characters in the string. The **SetTextCharacterExtra** function adds extra space to every character cell in the selected font, including the break character. (An application can use the **GetTextCharacterExtra** function to determine the current amount of extra space being added to the character cells; the default setting is zero.)

ABC spacing also allows an application to perform very accurate text alignment. For example, when an application right aligns a raster roman font without using ABC spacing, the advance width is calculated as the character width. This means the white space to the right of the glyph in the bitmap is aligned, not the glyph itself. By using ABC widths, applications have more flexibility in the placement and removal of white space when aligning text, because they have information that allows them to finely control intercharacter spacing.

### Rotating Text

Applications can rotate TrueType fonts at any angle. This is useful for labeling charts and other illustrations. The following example rotates a string in 10-degree increments around the center of the client area by changing the value of the lfEscapement member of the LOGFONT structure used to create the font:

```
RECT rc;  
int angle;  
HFONT hfnt, hfntPrev;  
LPSTR lpszRotate = "String to be rotated.";  
PLOGFONT plf = (PLOGFONT) LocalAlloc(LPTR, sizeof(LOGFONT));
```

```
lstrcpy(plf->lfFaceName, "Arial");  
plf->lfWeight = 700;
```

```
GetClientRect(hwnd, &rc);  
SetBkMode(hdc, TRANSPARENT);  
for (angle = 0; angle < 360; angle += 100) {  
    plf->lfEscapement = angle;  
    hfnt = CreateFontIndirect(plf);  
    hfntPrev = hfnt;  
    TextOut(hdc, rc.left + rc.right / 2, rc.bottom / 2, lpszRotate, lstrlen(lpszRotate));  
    DeleteFont(hfnt);  
    SetBkMode(hdc, OPAQUE);  
    LocalFree(LOCAL_MEM, plf);  
}
```

This example produces the following pattern:

The lfOrientation member of the LOGFONT structure is ignored by GDI, which currently, assumes that the values for lfEscapement and lfOrientation are identical.

### TrueType Font Functions and Structures

Some of the functions and structures that allow an application to take advantage of the extra functionality of TrueType are discussed elsewhere in this topic. This section describes some of the TrueType functions that are useful for applications that must take full advantage of the new font technology.

### Retrieving Character Outlines

Applications can use the GetGlyphOutline function to retrieve the outline of a glyph from a TrueType font. GetGlyphOutline returns the outline as a bitmap or as a series of polylines and splines.

When an application retrieves a glyph outline as a series of polylines and splines, the information is returned in a TPPOLYGONHEADER structure followed by as many TPPOLYCURVE structures as are required to describe the glyph. All points are returned as POINTFX structures and represent absolute positions, not relative moves. The starting point given by the ptfxStart member of the TPPOLYGONHEADER structure is the point at which the outline for a contour begins. The TPPOLYCURVE structures that follow can be either polyline records or spline records. Polyline records are a series of points; lines drawn between the points describe the outline of the character. Spline records represent the quadratic curves used by TrueType (that is, quadratic b-splines).

Each polyline and spline record contains as many sequential points as possible, to minimize the number of records returned.

The starting point given in the TTPOLYGONHEADER structure is always on the outline of the glyph. The specified point is both the starting point and the ending point for the contour.

A polyline record begins with the last point in the previous record (or with the starting point, for the first record in the contour). Each point in the record is on the glyph outline and can be connected simply by using straight lines.

A spline record begins with the last point in the previous record (or with the starting point, for the first record in the contour). For the first spline record, the starting point and the last point in the record are on the glyph outline. For all other spline records, only the last point is on the glyph outline. All other points in the spline records are off the glyph outline and must be rendered as the control points of b-splines.

The last spline or polyline record in a contour always ends with the contour's starting point. This ensures that every contour is closed.

Because b-splines require three points (one point that is off the glyph outline between two that are on the outline), applications must perform some calculations when a spline record contains more than one off-curve point.

For example, if a spline record contains three points (A, B, and C) and it is not the first record, points A and B are off the glyph outline. To interpret point A, an application can use the current position (which is always on the glyph outline) and the point on the glyph outline between points A and B. To find this point between A and B, the application can perform the following calculation:

$$M = A + (B - A)/2$$

The midpoint between consecutive off-outline points in a spline record is a point that is on the glyph outline, according to the definition of the spline format used in TrueType fonts. In preceding formula, M is the midpoint on the line between points A and B.

If the current position is designated by P, the two quadratic splines defined by this spline record are (P, A, M) and (M, B, C).

To render a TrueType character outline in GDI, an application must use both the polyline and the spline records. GDI can render polylines easily, but it does not support any spline formats. To use the spline records, an application must convert them into a series of polylines that approximate the spline.

The glyph outline returned by the GetGlyphOutline function is for a grid-fitted glyph. (A grid-fitted glyph has been modified so that its bitmap image conforms as closely as possible to the original design of the glyph.) If an application requires an unmodified glyph outline, it should request the glyph outline for a character in a font whose size is equal to the font's em units. (To create a font with this size, an application can set the lfHeight member of the LOGFONT structure to the negative of the value of the ntmSizeEM member of the NEWTEXTMETRIC structure.)

### Using Portable TrueType Metrics

Applications that use the TrueType font metrics can achieve a high degree of printer and document portability. Applications that must maintain compatibility with earlier versions of Windows can use the TrueType metrics, as can applications that are written specifically for Windows version 3.1.

### Design Widths

Design widths overcome most of the problems of device-dependent text introduced by physical devices. Design widths are a kind of logical width. Independent of any rasterization problems or scaling transformations, each glyph has a logical width and height. Composed to a logical page, each character in a string has a place independent of the physical device widths. Although a logical width implies that widths can be scaled linearly at all point sizes, this is not necessarily true for either nonportable or most TrueType fonts. At smaller point sizes, some glyphs are made wider relative to their height for better readability.

The characters in TrueType core fonts are designed against a 2048-by-2048 grid. The design width is the width of a character in these grid units. (TrueType supports any integer grid size up to 16,384 by 16,384; grid sizes that are integer powers of 2 scale faster than other grid sizes.)

The outline of a font is designed in notional units. The em square is the notional grid against which the font outline is fitted. (The otmEMSquare member of OUTLINETEXTMETRIC and the ntmSizeEM member of NEWTEXTMETRIC give the size of the em square in notional units.) When a font is created that has a point size (in device units) equal to the size of its em square, the ABC widths for this font are the desired design widths. For example, if the size of an em square is 1000 and the ABC widths of a character in the font are 150, 400, and 150, a character in this font that has a height of 10 in device units would have ABC widths of 1.5, 4, and 1.5, respectively. Since the MM\_TEXT mapping mode is most commonly used with fonts (and MM\_TEXT is equivalent to device units), this is a simple calculation.



Because of the high resolution of TrueType design widths, applications that use them must take into account the large numeric values that can be created.

#### Device vs. Design Units

Portable metrics in fonts are known as design units. To apply to a given device, design units must be converted to device units. An application can use the following formula to convert design units to device units:

$$\text{DeviceUnits} = (\text{DesignUnits}/\text{unitsPerEm}) * (\text{PointSize}/72) * \text{DeviceResolution}$$

The variables in this formula have the following meanings:

Variable	Description
DeviceUnits	Specifies the DesignUnits font metric converted to device units. This value is in the same units as the value given for DeviceResolution.
DesignUnits	Specifies the font metric to be converted to device units. This value could be any font metric, including the width of a character or the ascender value for an entire font.
unitsPerEm	Specifies the em square size for the font.
PointSize	Specifies size of the font in points. (One point equals 1/72 of an inch.)
DeviceResolution	Specifies number of device units (pixels) per inch. Typical values might be 300 for a laser printer or 96 for a VGA screen.

Note: This formula should not be used to convert device units back to design units. Device units are always rounded to the nearest pixel. The propagated round-off error can become very large, especially when an application is working with screen sizes.

#### Requesting Design-Unit Metrics

Font metrics for a physical font can be retrieved only after a font has been selected into a device context. When a font is selected into a device context, it is scaled for the device, which makes the font metrics specific to the device. To request design units, an application should create a logical font whose height is specified as -unitsPerEm. Applications can retrieve the value for unitsPerEm by calling the EnumFontFamilies function and checking the ntmSizeEM member of the NEWTEXTMETRIC structure.

#### Metrics for Portable Documents

The following table specifies the most important font metrics for applications that require portable documents and the functions that allow an application to retrieve them:

Function	Metric	Use
<u>EnumFontFamilies</u>	ntmSizeEM	Retrieving design metrics; conversion to device metrics
<u>GetCharABCWidths</u>	ABCWidths	Accurate placement of characters at the start and end of margins, picture boundaries, and other text breaks
<u>GetCharWidth</u>	AdvanceWidths	Placement of characters on a line. (This function is not new for Windows 3.1.)
<u>GetOutlineTextMetrics</u>	otmfsType	Font-embedding bits
	otmsCharSlopeRise	Y-component for slope of cursor for italic fonts
	otmsCharSlopeRun	X-component for slope of cursor for italic fonts
	otmAscent	Line spacing
	otmDescent	Line spacing
	otmLineGap	Line spacing
	otmpFamilyName	Font identification
	otmpStyleName	Font identification
otmpFullName	Font identification (typically, family and style name)	

The otmsCharSlopeRise, otmsCharSlopeRun, otmAscent, otmDescent, and otmLineGap members of the

OUTLINETEXTMETRIC structure are scaled or transformed to correspond to the current device mode and physical height (as given in the tmHeight member of the NEWTEXTMETRIC structure).

Font identification is important if the same font must be selected when a document is reopened or moved to a different system. The font mapper always selects the correct font when it is asked for by full name. The family and style names are needed in order to provide input to the standard font dialog box for proper placement of the selection bars.

The otmsCharSlopeRise and otmsCharSlopeRun values are used to produce a close approximation of the main italic angle of the font. For typical roman fonts, otmsCharSlopeRise is 1 and otmsCharSlopeRun is 0. For italic fonts, the values attempt to approximate the sine and cosine of the main italic angle of the font (in counterclockwise degrees past vertical); note that the italic angle for upright fonts is 0. Because these values are not expressed in design units, they should not be converted into device units.

The character placement and line spacing metrics allow an application to compute device-independent line breaks that are portable across screens, printers, typesetters, and even platforms. If all applications adopt these techniques, documents moved from one application to another will not reflow.

Device-independent page layout requires seven basic steps:

- 1 Normalize all design metrics to a common ultra-high resolution (UHR) value (for example, 65,536 DPI); this prevents round-off errors.
- 2 Compute line breaks based on UHR metrics and physical page width; this yields a starting point and an ending point of a line within the text stream.
- 3 Compute the device page width in device units (for example, pixels).
- 4 Fit each line of text into the device page width, using the line breaks computed in step 2.
- 5 Compute page breaks by using UHR metrics and the physical page length; this yields the number of lines per page.
- 6 Compute the line heights in device units.
- 7 Fit the lines of text onto the page, using the lines per page from step 5 and the line heights from step 6.

#### Panose Numbers

TrueType font files include Panose numbers, which applications can use to choose a font that closely matches their specifications. The Panose system classifies faces by 10 different attributes. These attributes are each rated on a scale. The resulting values are concatenated to produce a number. Given this number for a font and a mathematical metric to measure distances in the Panose space, an application can determine nearest neighbors. A PANOSE structure is part of the OUTLINETEXTMETRIC structure (whose values are filled in by calling the GetOutlineTextMetrics function).

#### Creating Customized Fonts

GDI keeps a system font table containing all the fonts that applications can use. GDI chooses a font from this table when an application calls the CreateFont or CreateFontIndirect function. There can be up to 253 entries in the system font table.

A font resource is a group of individual fonts representing characters in a given character set that have various combinations of heights, widths, and pitches. Applications can load font resources and add the fonts in the resource to the system font table by using the AddFontResource function. After a font resource has been added, the application can use the individual fonts in the resource. In other words, the CreateFont function takes the fonts into account when it tries to match a physical font with the specified logical font. (Fonts in the system font table are never directly accessible to an application. They are available only through the CreateFontIndirect and CreateFont functions, which return handles of the fonts, not memory addresses.)

An application can add a font resource to the system font table by using the AddFontResource function. To remove a font resource, an application can use the RemoveFontResource function.

Whenever an application adds or removes a font resource, it should inform all other applications of the change by sending a WM\_FONTCHANGE message to them. An application can use the following call to the SendMessage function to send the message to all windows:

```
SendMessage (HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
```

An application can use the GetProfileString function to retrieve a list of any fonts the user has used Control Panel to install. The application would use GetProfileString to search the [Fonts] section of the WIN.INI file.

An application can create font resources by creating font files and adding them as resources to a font resource file. To create a font resource file, an application should follow these steps:

- 1 Create the font files.
- 2 Create a resource-definition file for the font.
- 3 Create a dummy code module.
- 4 Create a module-definition file that describes the fonts and the devices that use the fonts.
- 5 Compile and link the sources.

A font resource file is an empty Windows dynamic-link library; it contains no code or data, but does contain resources. An application can add a font file to an empty library, along with such resources as icons, cursors, and menus, by using Microsoft Windows Resource Compiler (RC).

### Creating Font Files

An application can create raster font files by using Microsoft Windows Font Editor (FONTEDIT.EXE), as described in Microsoft Windows Programming Tools. (Font Editor cannot be used to generate vector or TrueType fonts.) The application can use any number, size, and type of font files in a font resource. In most cases, enough fonts should be included to reasonably satisfy most logical-font requests for the target device.

GDI can scale device-independent raster fonts by 1 to 8 times vertically and 1 to 5 times horizontally. GDI can also simulate bold, underlined, strikeout, and italic fonts. Font designers may choose to allow GDI to synthesize some sizes and properties of a font, rather than providing separate font files.

Font Editor modifies existing .FNT files; it cannot create font files from scratch. The Microsoft Windows 3.1 Software Development Kit (SDK) includes two .FNT files that font designers can load into Font Editor, modify, and save as customized fonts. The file named ATRM1111.FNT is a fixed-width font. The file named VGASYS.FNT is a variable-width font.

The Save As dialog box in Font Editor includes two File Format radio buttons. Font files saved in Font Editor 3.0 format can be used only in 386 enhanced mode. Font files saved in Font Editor 2.0 format can be used in all modes.

### Creating the Resource-Definition File for a Font

An application can add resources to a font file by adding one or more FONT statements to the resource-definition file. The resource-definition file can add .FNT files to a Windows library, a device driver, or a resource-only file that contains only icons, cursor, fonts, and other resources. Because font resources are available to all applications, they should not be added to application modules.

The FONT statement has the following form:

```
number FONT filename
```

One statement is required for each font file to be placed in the resource. The number must be unique, because it is used to identify the font later. The following is a typical resource-definition file for a font resource:

```
1 FONT FNTFIL01.FNT
2 FONT FNTFIL02.FNT
3 FONT FNTFIL03.FNT
4 FONT FNTFIL04.FNT
5 FONT FNTFIL05.FNT
6 FONT FNTFIL06.FNT
```

You can add fonts to modules that contain other resources by adding them to the existing resource-definition file. An application can have icon, menu, cursor, and dialog box definitions in the resource-definition file, as well as FONT statements.

### Creating a Dummy Code Module

A dummy code module provides the object file from which the font resource file is made. A developer can create the dummy code module by using the assembler and the Cmacros. The module's source file could look like this:

```
TITLE FONTRES - Stub file to build a .FON resource file

.xlist
include cmacros.inc
```

```
.list

sBegin CODE
db 0
sEnd CODE
end
```

Microsoft Segmented Executable Linker LINK version 4 allows empty code segments, but LINK versions 5.12 and later does not. The inclusion of "db 0" between sBegin and sEnd in the preceding example prevents an empty code segment.

The developer can assemble this source file by using the masm command. The object file that will be created will contain no code and no data, but it can be linked to an empty Windows library to which the font resources can be added.

Developers who build font files using version 6.0 of Microsoft Macro Assembler (ML) should use version 5.3 of the CMACROS.INC file (included with ML) instead of version 5.2 of the file, which is included with the SDK.

#### Creating a Module-Definition File

The module-definition file for the font resource must contain a LIBRARY statement that defines the resource name, a DESCRIPTION statement that describes the font resource characteristics, and a DATA statement. The module-definition file for a font resource should look like this:

```
LIBRARY FONTRES

DESCRIPTION 'FONTRES 133,96,72 : System, Terminal (Set #3)'

EXETYPE WINDOWS

STUB 'WINSTUB.EXE'
DATA NONE
```

The DESCRIPTION statement provides device-specific information about the font that is used to match a font with a given screen or printer. The following are the three possible formats for the DESCRIPTION statement in a font resource. In each case, the first characters in the description must be a single quote and the name of the library module (FONTRES):

```
DESCRIPTION 'FONTRES Aspect, LogPixelsX, LogPixelsY: Cmt' DESCRIPTION 'FONTRES
CONTINUOUSSCALING: Cmt' DESCRIPTION 'FONTRES DEVICESPECIFIC DeviceTypeGroup:
Cmt'
```

The first format specifies a font that was designed for a specific aspect ratio and logical pixel width and height, and can be used with any device having the same aspect and logical pixel dimensions. Aspect is the value  $(100 * \text{AspectY}) / \text{AspectX}$  rounded to an integer. The AspectX, AspectY, LogPixelsX, and LogPixelsY values are the same as the values given in the corresponding device's GDIINFO structure (values that are accessible by using the GetDeviceCaps function). You can specify more than one set of Aspect, LogPixelsX, and LogPixelsY values. The Cmt value is a comment. The following statements are examples:

```
DESCRIPTION 'FONTRES 133,96,72: System, Terminal (Set #3)'
DESCRIPTION 'FONTRES 200,96,48; 133,96,72; 83,60,72; 167,120,72: \
MS Sans Serif'
```

The second format specifies a continuous scaling font. This typically corresponds to vector fonts that can be drawn to any size and that do not depend on the aspect or logical pixel width of the output device. The following statement is an example:

```
DESCRIPTION 'FONTRES CONTINUOUSSCALING : Modern, Roman, Script'
```

The third format specifies a font that is specific to a particular device or group of devices. The DeviceTypeGroup can be DISPLAY or a list of device-type names--the same names an application might specify as the second parameter in a call to the CreateDC function. Following is an example of the third format:

```
DESCRIPTION 'FONTRES DISPLAY: HP 7470 plotters'
DESCRIPTION 'FONTRES DEVICESPECIFIC HP 7470A, HP 7475A: \
HP 7470 plotters'
```

Note: The maximum length of a DESCRIPTION line is 127 characters. Because GDI is capable of synthesizing attributes, such as bold, italic, and underline, the font designer need not create separate .FNT files for fonts with these attributes. Windows may use other fonts that do not correspond to the user's screen aspect ratio. These are generic raster fonts that are intended for output devices such as bitmap printers, which rely on the display driver to draw text.

#### Compiling and Linking a Font Resource File

The following makefile lists the commands required to compile and link a font resource file:

```
fontres.obj: fontres.asm
    masm fontres;

fontres.exe: fontres.def fontres.obj fontres.rc fontres.exe \
    fntfil01.fnt fntfil02.fnt fntfil03.fnt \
    fntfil04.fnt fntfil05.fnt fntfil06.fnt
    link fontres.obj, fontres.exe, NUL, /nod, fontres.def
    rc fontres.rc
    rename fontres.exe custom.fon
```

By convention, all raster font resource files have the .FON filename extension. The last line in the makefile renames the executable file to CUSTOM.FON.

#### Adding TrueType Fonts

Because Windows cannot directly interpret the native TrueType font file format, a file that mimics the standard .FON file (called a .FOT file) is required to make internal bookkeeping and enumeration easier. The `CreateScalableFontResource` function produces a .FOT file that points to the TrueType font file. Once this .FOT file is produced, Windows applications can use TrueType fonts transparently by using the `AddFontResource` and `RemoveFontResource` functions. Applications could also use the `CreateScalableFontResource` function to install special fonts for logos, icons, and other graphics.

#### Related Topics

## Window Management (3.1)

The following topics describe the functions in the Microsoft Windows operating system that process messages; create, move, or alter a window; or create system output. These functions constitute the window manager interface.

Caret functions

Cursor functions

Dialog box procedures

Hook functions

Message functions

Painting functions

Property functions

Rectangle functions

Scrolling functions

Window-creation functions

## Messages

Messages are the input to an application. They represent events that the application may need to respond to. A message is a structure that contains a message identifier and message parameters. The content of the parameters varies with the message type.

### Generating and Processing Messages

Windows generates an input message for each input event, such as when the user moves the mouse or presses a key. Windows collects input messages in a systemwide message queue and then places the messages, as well as timer and paint messages, in an application message queue. An application message queue is a first in, first out queue. Timer and paint messages are exceptions to the first in, first out rule; these messages are held in an application's message queue until the application has processed all other messages. Windows places messages that belong to a specific application in that application's message queue. The application then reads the messages by using the `GetMessage` function and dispatches them to the appropriate window procedure by using the `DispatchMessage` function.

Windows sends some messages directly to the window procedure in the appropriate application instead of placing the messages in the application's message queue. Such messages are called unqueued messages. Typically, an unqueued message is any message that affects the window only. The `SendMessage` function sends messages directly to a window procedure. For more information about window procedures, see Window Procedures.

For example, the `CreateWindow` function directs Windows to send a `WM_CREATE` message to a window procedure of an application and to wait until the window procedure has processed the message. Windows sends this message directly to the window procedure and does not place it in the application's message queue.

Although Windows generates most messages, an application can create its own messages and place them in its own message queue or that of another application.

An application typically uses the `GetMessage` function in a loop within its `WinMain` function to remove messages from the application's message queue. This loop is called the main message loop. The `GetMessage` function searches an application's message queue and, if any messages exist, returns the top message in the queue. If the message queue is empty, `GetMessage` waits for a message to be placed in the queue. While waiting, `GetMessage` relinquishes control to Windows, allowing other applications to take control and process their own messages.

Once an application's `WinMain` function has retrieved a message from the application's message queue, it can dispatch the message to a window procedure by using the `DispatchMessage` function. This function directs Windows to call the window procedure of the window associated with the message, and then passes the content of the message as function arguments. The window procedure can then process the message and carry out any requested changes to the window. When the window procedure returns, Windows returns control to the main message loop in the `WinMain` function. The main message loop can then retrieve the next message from the queue.

Note: Unless noted otherwise, Windows can send messages in any sequence. An application should not rely on receiving messages in a particular order.

Windows generates a message each time the user presses a key. The message contains a virtual-key code that defines which key was pressed, but does not define the character value of that key. To retrieve the character value, the main message loop in the `WinMain` function must translate the virtual-key message by using the `TranslateMessage` function. This function puts another message with an appropriate character value in the application's message queue. The message can then be dispatched to a window procedure.

### Translating Messages

In general, a `WinMain` function should use the `TranslateMessage` function to translate every message, not just virtual-key messages. Although `TranslateMessage` has no effect on other types of messages, it guarantees that keyboard input is translated correctly.

The following example illustrates the typical main message loop that a `WinMain` function uses to retrieve messages from the application's message queue and dispatch them to the application's window procedures:

```
int PASCAL WinMain(HINSTANCE hinstCurrent, HINSTANCE hinstPrevious,
    LPSTR lpzCmdLine, int nCmdShow)
{
    MSG msg;
```

```

.
.
.

    while (GetMessage(&msg, NULL, 0, 0)) {
TranslateMessage(&msg); /* translates virtual key codes */
DispatchMessage(&msg); /* dispatches message to window */
    }
    return (int) msg.wParam; /* return value of PostQuitMessage */
}

```

An application that uses accelerator keys must load an accelerator table from the resource-definition file by using the LoadAccelerators function and then translate keyboard messages into accelerator-key messages by using the TranslateAccelerator function.

The main message loop for an application that uses accelerator keys should have the following form:

```

while (GetMessage(&msg, NULL, 0, 0)) {
    if (TranslateAccelerator(hwnd, haccel, &msg) == 0) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
}
return (int) msg.wParam;

```

The TranslateAccelerator function must appear before the standard TranslateMessage and DispatchMessage functions. Furthermore, because TranslateAccelerator automatically dispatches the accelerator-key message to the appropriate window procedure, the TranslateMessage and DispatchMessage functions should not be called if TranslateAccelerator returns a nonzero value.

### Examining Messages

An application can use the PeekMessage function to examine its message queue for specific messages without removing them from the queue. The function returns a nonzero value if a message exists in the queue and lets the application retrieve the message and process it without going through the application's main message loop.

Typically, an application uses PeekMessage to check periodically for messages when the application is carrying out a lengthy operation, such as processing input and output. For example, this function can be used to check for messages that end the operation. PeekMessage also gives the application a chance to yield control if no messages are present, because PeekMessage can yield if no messages are in the message queue.

### Sending Messages

The SendMessage and PostMessage functions let applications pass messages to their windows or to the windows of other applications. The PostAppMessage function is a variation on PostMessage that posts a message using the application's module handle rather than a window handle.

The PostMessage function directs Windows to post a message--that is, place the message in an application's message queue. The PostMessage function immediately returns control to the calling application, and any action to be carried out as a result of the message does not occur until the message is read from the queue.

The SendMessage function directs Windows to send a message directly to the given window procedure, bypassing the application's message queue. Windows does not return control to the calling application until the window procedure that receives the message processes the message or returns control as a result of a call to the ReplyMessage function.

When an application transmits a message, it must do so by calling SendMessage if the application relies on the return value of a message. The return value of SendMessage is the same as the value returned by the window procedure that processed the message. PostMessage returns immediately after sending the message, so its return value is only a Boolean value indicating whether the message was successfully placed in the queue and does not indicate how the message was processed.

### Avoiding Message Deadlocks

An application can create a deadlock condition in Windows if it yields control while processing a message sent from another application (or by Windows on behalf of another application) by using the SendMessage function.



Typically, a task that calls SendMessage to send a message to another task does not continue running until the window procedure that receives the message returns. When the task that receives the message yields control, the sending task cannot continue to run and to process messages because it is waiting for SendMessage to return, resulting in a message deadlock.

The application processing the message does not have to yield explicitly to cause the problem. Calling any one of the following functions can result in the application yielding control:

- DialogBox
- DialogBoxIndirect
- DialogBoxIndirectParam
- DialogBoxParam
- GetMessage
- MessageBox
- PeekMessage
- Yield

Before calling any of these functions while processing a message, a window procedure should first call the InSendMessage function to find out whether the message was sent by the SendMessage function from another application. If InSendMessage returns a nonzero value, the window procedure must call the ReplyMessage function before calling any function that yields control.

## Creating and Managing Windows

This section describes how to create, destroy, modify, and obtain information about windows.

### Window Classes

A window class is a set of attributes that defines how a window looks and behaves. Before an application can create and use a window, a window class must have been created and registered for that window. An application registers a class by filling a `WNDCLASS` structure and passing a pointer to the structure to the `RegisterClass` function. Any number of window classes can be registered. Once a class has been registered, Windows lets the application create any number of windows belonging to that class. The registered class remains available until it is deleted or the application closes.

Although the complete window class consists of many elements, Windows requires only that an application supply a class name, the address of the window procedure that will process all messages sent to windows belonging to this class, and an instance handle identifying the application that registered the class. The other elements of the window class define default attributes for windows of the class, such as the shape of the cursor and the content of the menu for the window.

There are three types of window classes: system global classes, application global classes, and application local classes. These types differ in scope and in when and how they are created and destroyed.

### System Global Classes

Windows creates system global classes when it starts. These classes are available for use by all applications at all times. Because Windows creates system global classes on behalf of all applications, an application cannot create or destroy any of these classes. System global classes include edit-control and list-box control classes.

### Application Global Classes

An application or (more likely) a dynamic-link library (DLL) creates an application global class by specifying the `CS_GLOBALCLASS` style for the class. Once created, it is globally available to all applications within the system. Typically, a DLL creates an application global class so that applications that call the DLL can use the class. Windows destroys an application global class when the application that created it closes or the DLL that created it is unloaded. For this reason, it is essential that all applications destroy all windows using that class before the application that created the class closes or the DLL that created the class is unloaded. Use the `UnregisterClass` function to remove an application global class and free the storage associated with it.

### Application Local Classes

An application local class is any window class created by an application for its exclusive use. This is the more common type of class created by an application. Use the `UnregisterClass` function to remove an application local class and free the storage associated with it.

### How Windows Locates a Class

When an application creates a window with a specified class, Windows uses the following procedure to find the class:

- 1 Windows searches for a local class of the specified name.
- 2 If Windows does not find a local class with the name, it searches the application global class list.
- 3 If Windows does not find the name in the application global class list, it searches the system global class list.

This procedure is used for all windows created by the application, including windows created by Windows on the application's behalf, such as dialog boxes. It is possible, then, to override system global classes without affecting other applications.

### Class Ownership

Windows determines class ownership from the `hInstance` member of the `WNDCLASS` structure passed to the `RegisterClass` function when the application or DLL registers the class. For Windows DLLs, the `hInstance` member must be the instance handle of the DLL. When the application that registered the class closes or the DLL that registered the class is unloaded, the class is destroyed. For this reason, all windows using the class must be destroyed before the application closes or the DLL is unloaded.

### Registering a Window Class

When Windows registers a window class, it copies the attributes into its own memory area. Windows uses these internally stored attributes when an application refers to the window class by name; it is not necessary for the application that originally registers the class to keep the structure available.

#### Shared Window Classes

An application must not share its registered classes with other applications. Some information in a window class, such as the address of the window procedure, is specific to a given application and cannot be used by other applications. However, applications can share an application global class. For more information, see Application Global Classes.

Although an application must not share one of its registered classes with other applications, different instances of the same application can share a registered class. Once a window class has been registered by an application, it is available to all subsequent instances of that application. This means that new instances of an application do not need to, and should not, register window classes that have been registered by previous instances.

#### Predefined Window Classes

Windows provides several predefined system-global window classes. These classes define special control windows that carry out common input tasks, such as letting the user direct scrolling, type text, and select from a list of names. The predefined window classes are available to all applications and can be used any number of times to create any number of control windows.

#### Elements of a Window Class

The elements of a window class define the default behavior of windows created from that class. The application that registers a window class assigns elements to the class by setting appropriate members in a WNDCLASS structure and passing the structure to the RegisterClass function. An application can retrieve information about a given window class with the GetClassInfo function. The window class elements are as follows:

Element	Purpose
Class name	Distinguishes the class from other registered classes.
Window-procedure address	Points to the function that processes all messages that are sent to windows in the class, and defines the behavior of the window.
Instance handle	Identifies the application or DLL that registered the class.
Class cursor	Defines the shape of the cursor when the cursor is in a window of the class.
Class icon	Defines the shape of the icon Windows displays when a window belonging to the class is minimized.
Class background brush	Defines the color and pattern Windows uses to fill the client area when the window is opened or painted. If this parameter is set to NULL, the window must paint its own background whenever it receives the <u>WM_ERASEBKGD</u> message.
Class menu	Specifies the default menu used for any window belonging to the class that does not explicitly define a menu.
Class styles	Defines how to update the window after moving or resizing, how to process double-clicks of the mouse, how to allocate space for the display context, and other aspects of the window.
Class extra	Specifies the amount of extra memory, in bytes, that Windows should reserve at the end of the <u>WNDCLASS</u> structure. Windows initializes this memory to zero.
Window extra	Specifies the amount of extra memory, in bytes, that Windows should reserve at the end of any window structure an application creates that has this class. Windows initializes this memory to zero.

The following sections describe the elements of a window class and explain the default values for these elements if no explicit value is given when the class is registered.

#### Class Name

Every window class needs a class name. The class name distinguishes one class from another. An application assigns a class name to the class by setting the lpszClassName member of the WNDCLASS structure to the address of a null-terminated string that specifies the name.

In the case of an application global class, the class name must be unique to distinguish it from other application global classes. If an application registers another application global class with the name of an existing application global class, the `RegisterClass` function returns zero, indicating failure. The conventional method for ensuring this uniqueness is to include the application name in the name of the application global class.

The class name must be unique among all the classes registered by an application. An application cannot register an application local class and an application global class with the same class name.

#### Window-Procedure Address

Every class needs a window-procedure address. The address defines the entry point of the window procedure that is used to process all messages for windows in the class. Windows passes messages to the procedure when it requires the window to carry out tasks, such as painting its client area or responding to input from the user. An application assigns a window-procedure to a class by copying its address to the `lpfnWndProc` member of the `WNDCLASS` structure. The window procedure must be exported in the module-definition (.DEF) file. For more information about exporting functions, see the `EXPORTS` topic.

#### Instance Handle

Every window class needs an instance handle to identify the application or DLL that registered the class. As a multitasking system, Windows lets several applications or DLLs run at the same time, so it needs instance handles to keep track of all applications and DLLs. Windows assigns a unique handle to each copy of a running application or DLL.

Multiple instances of the same application or DLL all use the same code segment, but each has its own data segment. Windows uses an instance handle to identify the data segment that corresponds to a particular instance of an application or DLL.

Windows passes an instance handle to an application or DLL when the application first begins operation. The application or DLL assigns this instance handle to the class by copying it to the `hInstance` member of the `WNDCLASS` structure.

#### Class Cursor

The class cursor defines the shape of the cursor when the cursor is in the client area of a window in the class. Windows automatically sets the cursor to the given shape as soon as the cursor enters the window's client area, and ensures that the cursor keeps that shape while it remains in the client area. To assign a cursor shape to a window class, an application typically loads a predefined cursor shape by using the `LoadCursor` function, and then assigns the returned cursor handle to the `hCursor` member of the `WNDCLASS` structure. Alternatively, you can use Microsoft Image Editor (IMAGEDIT.EXE) to create your own custom cursor, and use Microsoft Windows Resource Compiler (RC) to add the cursor as a resource to your application's executable file. The application can then use the `LoadCursor` function to load the custom cursor from the application's resources.

Windows does not require a class cursor. If an application sets the `hCursor` member of the `WNDCLASS` structure to `NULL`, a class cursor is not defined. Windows assumes that the window will set the cursor shape each time the cursor moves into the window. A window can set the cursor shape by calling the `SetCursor` function whenever the window receives the `WM_MOUSEMOVE` message.

#### Class Icon

The class icon defines the shape of the icon used when the window of the given class is minimized. To assign an icon to a window class, an application typically loads the icon from the application's resources by using the `LoadIcon` function, and then assigns the returned icon handle to the `hIcon` member of the `WNDCLASS` structure.

Windows does not require that a window class have a class icon. If an application sets the `hIcon` member of the `WNDCLASS` structure to `NULL`, a class icon is not defined. In this case, Windows sends the `WM_ICONERASEBKGND` message to a window of the class whenever the window must paint the background of the icon. If the window does not process the `WM_ICONERASEBKGND` message, Windows draws an image of the contents of the window's client area onto the icon when it is minimized.

#### Class Background Brush

A class background brush is the brush used to prepare the client area of a window for subsequent drawing by the application. Windows uses the brush to fill the client area with a solid color or pattern, thereby removing all previous images from that location whether they belonged to the window or not. Windows notifies a window that its background needs to be painted by sending the `WM_ERASEBKGND` message to the window.

To assign a background brush to a class, an application can create a brush by using the appropriate functions from the graphics device interface (GDI) and then assign the returned brush handle to the `hbrBackground` member of the `WNDCLASS` structure.

Instead of creating a brush, an application can use a standard system color by setting the `hbrBackground` member to one of the standard system color values.

To use a standard system color, the application must increase the background-color value by one. For example, `COLOR_BACKGROUND + 1` is the system background color.

### Class Menu

A class menu defines the default menu to be used by the windows in the class if no explicit menu is given when the windows are created. A menu is a list of commands from which a user can select actions for the application to carry out. To assign a menu to a class, an application sets the `lpszMenuName` member of the `WNDCLASS` structure to the address of a null-terminated string that specifies the resource name of the menu. The menu is assumed to be a resource in the given application. Windows automatically loads the menu when it is needed. Note that if the menu resource is identified by an integer and not by a name, the application can set the `lpszMenuName` member to that integer value by applying the `MAKEINTRESOURCE` macro before assigning the value.

Windows does not require a class menu. If an application sets the `lpszMenuName` member of the `WNDCLASS` structure to `NULL`, Windows assumes that the windows in the class have no menu bars. Even if no class menu is given, an application can still define a menu bar for a window when it creates the window.

Windows does not allow menu bars with child windows. If a menu is given for a class and a child window of that class is created, the menu is ignored. For more information about menus, see `Menus`.

### Class Styles

The class styles define additional elements of the window class. Two or more styles can be combined by using the bitwise OR (`|`) operator. The class styles are as follows:

Style	Description
<code>CS_BYTEALIGNCLIENT</code>	Aligns the window's client area on a byte boundary (in the x direction).
<code>CS_BYTEALIGNWINDOW</code>	Aligns the window on a byte boundary (in the x direction).
<code>CS_CLASSDC</code>	Allocates one display context to be shared by all windows in the class. For more information about device contexts, see <code>Class and Private Display Contexts</code> , and <code>Display Context Types</code> .
<code>CS_DBLCLKS</code>	Sends double-click messages to the window procedure.
<code>CS_GLOBALCLASS</code>	Specifies that the window class is an application global class. An application global class is created by an application or DLL and is available to all applications. The class is destroyed when the application or DLL that created the class closes; it is essential, therefore, that all windows created with the application global class be closed before the application or DLL closes.
<code>CS_HREDRAW</code>	Requests that the entire client area be redrawn if a movement or size adjustment changes the width of the client area.
<code>CS_NOCLOSE</code>	Inhibits the Close command on the System menu (sometimes referred to as the Control menu).
<code>CS_OWDC</code>	Allocates a unique display context for each window in the class. For more information about device contexts, see <code>Class and Private Display Contexts</code> , and <code>Display Context Types</code> .
<code>CS_PARENTDC</code>	Gives the parent window's display context to the child windows. For more information about device contexts, see <code>Class and Private Display Contexts</code> , and <code>Display Context Types</code> .
<code>CS_SAVEBITS</code>	Saves, as a bitmap, the portion of the screen image that is obscured by a window; Windows uses the saved bitmap to re-create the screen image when the window is removed. Windows displays the bitmap at its original location and does not send <code>WM_PAINT</code> messages to windows that had been obscured by the window if the memory used by the bitmap has not been discarded and if other screen actions have not invalidated the stored image.

CS\_VREDRAW

Requests that the entire client area be redrawn if a movement or size adjustment changes the height of the client area.

To assign a style to a window class, an application assigns the style value to the style member of the WNDCLASS structure.

#### Internal Data Structures

Windows maintains internal data structures for each window class and window. These structures are not directly accessible to applications but can be examined and modified by using the following functions:

- GetClassInfo
- GetClassLong
- GetClassName
- GetClassWord
- GetWindowLong
- GetWindowWord
- SetClassLong
- SetClassWord
- SetWindowLong
- SetWindowWord

#### Window Subclassing

A subclass is a window or set of windows that belong to the same window class, and whose messages are intercepted and processed by another window procedure (or procedures) before being passed to the class window procedure.

To create the subclass, the SetWindowLong function is used to change which window procedure is associated with a particular window, causing Windows to call the new window procedure instead of the previous one. An application must call the CallWindowProc function to pass to the previous window procedure any messages not processed by the new window procedure. This allows Windows to create a chain of window procedures. The application can retrieve the address of the previous window procedure by using the GetWindowLong function before using the SetWindowLong function.

Similarly, the SetClassLong function changes which window procedure is associated with a window class. Any window that is subsequently created with that class will be associated with the replacement window procedure for that class, as will the window whose handle is passed to SetClassLong. Other existing windows that were previously created with the class are not affected, however.

When an application subclasses a window or class of windows, it must export the replacement window procedure in its module-definition file, call the MakeProcInstance function to create the address of the procedure, and pass the address to the SetWindowLong or SetClassLong function.

#### Redrawing the Client Area

When a window is moved, Windows automatically copies the contents of the client area to the new location. This saves time because a window does not have to recalculate and redraw the contents of the client area as part of the move. If the window moves and changes size, Windows copies only as much of the previous client area as is needed to fill the new location. If the window increases in size, Windows copies the entire client area and sends a WM\_PAINT message to the window to fill in the newly exposed areas.

When a window is moved, Windows assumes the contents of the client area remain valid and can be copied without modification to the new location. For some windows, however, the contents of the client area are not valid after a move, especially if the move includes a change in size. For example, a clock application whose window must always contain the complete image of the clock has to redraw the window anytime the window changes size, and has to update the time after the move. To redraw the entire client area instead of copying the previous contents each time a window changes size, a window should specify the CS\_VREDRAW and CS\_HREDRAW styles in the window class.

#### Class and Private Display Contexts

A display context is a special set of values that applications use for drawing in the client area of their windows. Windows requires a display context for each window on the system display but allows some flexibility in how that display context is stored and treated by the system.

If no display-context style is explicitly given, Windows assumes that each window will use a display context retrieved from a pool of contexts maintained by Windows. In such cases, each window must retrieve and initialize the display context before painting, and then free it after painting.

To avoid retrieving a display context each time it needs to paint inside a window, an application can specify the `CS_OWNDC` style for the window class. This class style directs Windows to create a private display context--that is, to allocate a unique display context for each window in the class. The application need only retrieve the context once, and then use it for all subsequent painting. Although the `CS_OWNDC` style is convenient, it must be used carefully because each display context uses a significant amount of system resources.

By specifying the `CS_CLASSDC` style, an application can have some of the convenience of a private display context without allocating a separate display context for each window. The `CS_CLASSDC` style directs Windows to create a single class display context--that is, one display context to be shared by all windows in the class. An application need only retrieve the display context for a window; as long as no other window in the class retrieves that display context, the window can continue to use the context.

Similarly, by specifying the `CS_PARENTDC` style, an application can create child windows that inherit the device context of their parent.

## Window Procedures

A window procedure processes all messages sent to all windows in a given class. Windows sends messages to a window procedure when it receives input from the user that is intended for the given window, or when it needs the procedure to carry out some action on its window, such as painting inside the client area.

A window procedure receives the following types of messages:

- Input messages from the keyboard, mouse or other pointing device, and timer
- Requests for information, such as a request for the window title
- Reports of changes made to the system by other windows, such as a change to the WIN.INI file
- Messages that give the window procedure an opportunity to modify the standard system response to certain actions, such as an opportunity to adjust a menu before it is displayed
- Requests to carry out some action on its window or client area, such as a request to update the client area
- Information about its status in relation to other windows, such as its losing access to the keyboard or becoming the active window

Most of the messages a window procedure receives are from Windows, but it can also receive messages from other windows, including windows it owns. These messages can be requests for information or notification that a given event has occurred within another window.

A window procedure continues to receive messages from the system and possibly other windows in the system until the window procedure, the window procedure of a parent window, or the system destroys the window. Even while the window is in the process of being destroyed, the window procedure receives additional messages that give it the opportunity to carry out any cleanup tasks before terminating. These messages include `WM_CLOSE`, `WM_DESTROY`, `WM_QUERYENDSESSION`, and `WM_ENDSESSION`. But once the window is destroyed, no more messages are passed to the procedure for that particular window. If there is more than one window of the class, however, the window procedure continues to receive messages for the other windows until they, too, are destroyed.

A window procedure defines how all windows of a given window actually behave; that is, it defines what response the windows make to commands from the user or system. The window procedure must examine messages it receives from the system and determine what action, if any, to take. For example, if the user clicks the scroll bar, the window procedure may scroll the contents of the client area. Windows passes information that affects a window and provides some tools to carry out tasks, such as drawing and scrolling, but the window procedure must carry out each actual task.

A window procedure can also choose not to respond to a given message. If it does not respond, the procedure must pass the message to the `DefWindowProc` function to give the system the opportunity to respond. This function carries out default actions based on the given message and its parameters. Many messages, especially nonclient-area messages, must be processed, so the `DefWindowProc` function is required in all window procedures.

A window procedure also receives messages that are really intended to be processed by the system. These messages, called nonclient-area messages, inform the procedure either that the user has carried out some action in a nonclient area of the window, such as clicking the title bar, or that some information about the window is required by the system to carry out an action, such as to move or adjust the size of the window. Although Windows passes these messages to the window procedure, the procedure should pass them to the `DefWindowProc` function and not attempt to process them. In any case, the window procedure must not ignore the message or return without passing it to `DefWindowProc`.

## Window Messages

A window message is a set of values that Windows sends to a window procedure to provide input to the window or request the window to carry out some action. Windows includes a wide variety of messages that it or applications can send to a window procedure. Most messages are sent to a window as a result of a given function being executed or as a result of input from the user.

Every message consists of four values: a handle that identifies the window, a message identifier, a 16-bit message-specific value, and a 32-bit message-specific value. These values are passed as individual parameters to the window procedure. The window procedure then examines the message identifier to determine what response to make and how to interpret the 16- and 32-bit values.

A window procedure must use the Pascal calling convention. The following illustrates the window procedure syntax:

```
LONG FAR PASCAL WndProc(hwnd, wParam, lParam)
HWND hwnd;
WORD wParam;
WORD lParam;
DWORD lParam;
```

The `hwnd` parameter identifies the window receiving the message; the `wParam` parameter is the message identifier; the `lParam` parameter is 16 bits of additional message-specific information; and `lParam` is 32 bits of additional message-specific information. The window procedure must return a 32-bit value that indicates the result of message processing. The possible return values depend on the actual message sent.

Windows expects to make an intersegment call to the window procedure, so the procedure must be declared with the `FAR` attribute. The window-procedure name must be exported by including it in an `EXPORTS` statement in the application's module-definition file.

#### Default Window Procedure

The `DefWindowProc` function is the default message processor for window procedures that do not or cannot process some of the messages sent to them. For most window procedures, the `DefWindowProc` function carries out most, if not all, processing of nonclient-area messages. These are the messages that signify actions to be carried out on parts of the window other than the client area. The messages that `DefWindowProc` processes and the default actions for each are as follows:

Message	Default action
<code>WM_ACTIVATE</code>	Activates or deactivates a window.
<code>WM_CANCELMODE</code>	Cancels internal processing of standard scroll bar input, cancels internal menu processing, and releases mouse capture.
<code>WM_CHARTOITEM</code>	Returns -1.
<code>WM_CLOSE</code>	Calls the <code>DestroyWindow</code> function.
<code>WM_CTLCOLOR</code>	Sets the background and text color and returns a handle of the brush used to fill the control background.
<code>WM_DRAWITEM</code>	Draws the focus rectangle for an owner-drawn list box item.
<code>WM_ERASEBKGD</code>	Fills the client area with the color and pattern specified by the class brush, if any.
<code>WM_GETTEXT</code>	Copies the window title into a specified buffer.
<code>WM_GETTEXTLENGTH</code>	Returns the length, in bytes, of the window title.
<code>WM_ICONERASEBKGD</code>	Fills the icon's client area with the window's background brush.
<code>WM_KEYUP</code>	Sends a <code>WM_SYSCOMMAND</code> message to the top-level window if the <code>F10</code> key or the <code>ALT</code> key was released. The <code>wParam</code> parameter of the message is set to <code>SC_KEYMENU</code> .
<code>WM_MOUSEACTIVATE</code>	Sends the <code>WM_MOUSEACTIVATE</code> response to the parent window. The parent determines whether to activate the child window.
<code>WM_NCACTIVATE</code>	Activates or deactivates the window and draws the icon or title bar to show the new state.
<code>WM_NCCALCSIZE</code>	Computes the size of the client area.
<code>WM_NCCREATE</code>	Initializes standard scroll bars, if any, and sets the default title for the window.
<code>WM_NCDESTROY</code>	Frees any space internally allocated for the window title.
<code>WM_NCHITTEST</code>	Finds out what part of the window the mouse is in.



WM_NCLBUTTONDBLCLK	Tests the given point to find out the location of the mouse and, if necessary, generates additional messages.
WM_NCLBUTTONDOWN	Finds out whether the left mouse button was pressed while the mouse was in the nonclient area of a window.
WM_NCLBUTTONUP	Tests the given point to find out the location of the mouse and, if necessary, generates additional messages.
WM_NCMOUSEMOVE	Tests the given point to find out the location of the mouse and, if necessary, generates additional messages.
WM_NCPAINT	Paints the nonclient areas of the window.
WM_PAINT	Validates the current update region, but does not paint the region.
WM_QUERYENDSESSION	Returns TRUE.
WM_QUERYOPEN	Returns TRUE.
WM_SETCURSOR	Displays the appropriate mouse cursor, based on the position of the cursor.
WM_SETREDRAW	Forces an immediate update of information about the clipping region of the complete window.
WM_SETTEXT	Sets and displays the window title.
WM_SHOWWINDOW	Opens or closes a window.
WM_SYSCCHAR	Generates a <u>WM_SYSCOMMAND</u> message for menu input.
WM_SYSCOMMAND	Carries out the requested system command.
WM_SYSKEYDOWN	Examines the given key and generates a <u>WM_SYSCOMMAND</u> message if the key is either TAB or ENTER.
WM_SYSKEYUP	Sends a <u>WM_SYSCOMMAND</u> message to the top-level window if the F10 key or the ALT key was released. The wParam parameter of the message is set to SC_KEYMENU.
WM_VKEYTOITEM	Returns -1.
WM_WINDOWPOSCHANGED	Sends the <u>WM_SIZE</u> and <u>WM_MOVE</u> messages to the window.
WM_WINDOWPOSCHANGING	Sends the <u>WM_GETMINMAXINFO</u> message to the window if the window has the <u>WS_OVERLAPPED</u> or <u>WS_THICKFRAME</u> style.

## Window Styles

Windows provides several different window styles that can be combined to form different kinds of windows. The styles are used in the CreateWindow function when the window is created.

### Overlapped Windows

An overlapped window is always a top-level window. In other words, an overlapped window never has a parent window. It has a client area, a border, and a title bar. It can also have a System menu, Minimize and Maximize buttons, scroll bars, and a menu, if these items are specified when the window is created. For a window used as a main interface, the System menu and Minimize and Maximize buttons are strongly recommended.

Every overlapped window can have a corresponding icon that Windows displays when the window is minimized. A minimized window is not destroyed. It can be restored to its previous size and position. An application minimizes a window to save screen space when several windows are open at the same time.

An application creates an overlapped window by using the WS\_OVERLAPPED or WS\_OVERLAPPEDWINDOW style with the CreateWindow function. An overlapped window created with the WS\_OVERLAPPED style always has a title bar and a border. The WS\_OVERLAPPEDWINDOW style creates an overlapped window with a title bar, a thick-frame border, a System menu, and Minimize and Maximize buttons.

### Owned Windows

An owned window is a special type of overlapped window. Every owned window must be owned by an overlapped window. Being owned forces several constraints on a window:

- An owned window is always in front of its owner when the windows are in z-order. Attempting to move the owner--that is, on an imaginary z-axis extending in front of the owned window from the screen toward the user--causes the owned window also to change position to ensure that it will always be in front of its owner.
- Windows automatically destroys an owned window when it destroys the window's owner.

- An owned window is hidden when its owner is minimized.

An application creates an owned window by specifying the owner's window handle as the `hWndParent` parameter of the `CreateWindow` function when creating a window that has the `WS_OVERLAPPED` style.

Dialog boxes are owned windows by default. The function that creates the dialog box receives the handle of the owner window as its `hWndParent` parameter.

#### Pop-up Windows

Pop-up windows are another special type of overlapped window. The main difference between a pop-up window and other overlapped windows is that an overlapped window always has a title bar, whereas the title bar is optional for a pop-up window. Like other overlapped windows, pop-up windows can be owned.

You create a pop-up window by using the `WS_POPUP` window style with the `CreateWindow` function. An application can use the `ShowWindow` function to open or close a pop-up window.

#### Child Windows

A child window is a window that is confined to the client area of a parent window. Child windows are typically used to divide the client area of a parent window into different functional areas.

You create a child window by using the `WS_CHILD` window style with the `CreateWindow` function. An application can use the `ShowWindow` function to show or hide a child window.

Every child window must have a parent window. The parent window can be an overlapped window, a pop-up window, or even another child window. The parent window relinquishes a portion of its client area to the child window, and the child window receives all input from this area. The window class does not have to be the same for each of the child windows of the parent window. This means an application can fill a parent window with child windows that look different and carry out different tasks.

A child window has a client area, but it does not have any other features unless these are explicitly requested. An application can request a border, title bar, Minimize and Maximize buttons, and scroll bars for a child window. In most cases, the application designs its own features for the child window.

Although it is not required, every child window should have a unique integer identifier. The identifier, given in the `hmenu` parameter of the `CreateWindow` function in place of a menu, helps identify the child window when its parent window has other child windows. The child window should use this identifier in any messages it sends to the parent window. This is the way a parent window with multiple child windows can identify which child window is sending the message. Child windows that share the same parent window are sibling windows.

Windows always positions the child window relative to the upper-left corner of the parent window's client area. The coordinates are always client coordinates. (For information about mapping, see Graphics Device Interface Overview.) If all or part of a child window is moved outside the visible portion of the parent window's client area, the child window is clipped; that is, the portion outside the parent window's client area is not displayed.

A child window is an independent window that receives its own input and other messages. Input intended for a child window goes directly to the child window and is not passed through the parent window. The only exception is if input to the child window has been disabled by the `EnableWindow` function. In this case, Windows passes any input that would have gone to the child window to the parent window instead. This gives the parent window an opportunity to examine the input and enable the child window, if necessary.

Actions that affect the parent window can also affect the child window, as follows:

Parent window	Child window
Shown	Shown after the parent window is shown.
Hidden	Hidden before the parent window is hidden. A child window can be visible only when the parent window is visible.
Destroyed	Destroyed before the parent window is destroyed.
Moved	Moved with the parent window's client area. The child window is responsible for painting after the move.
Increased in size or maximized	Paints any portions of the parent window that have been exposed as a result of the increased size of the client area.

Windows does not automatically clip a child window from the parent window's client area. This means the parent window draws over the child window if it carries out any drawing in the same location as the child

window. Windows does clip the child window from the parent window's client area if the parent window has a WS\_CLIPCHILDREN style. If the child window is clipped, the parent window cannot draw over it.

A child window can overlap other child windows in the same client area. Sibling windows can draw in each other's client area unless one child window has a WS\_CLIPSIBLINGS style. If the application specifies this style for a child window, any portion of that child's sibling window that lies within this window is clipped.

If a window has either the WS\_CLIPCHILDREN or WS\_CLIPSIBLINGS style, a slight loss in performance occurs.

Each window takes up system resources, so an application should not use child windows indiscriminately. For optimum performance, an application that needs to logically divide its main window should do so in the window procedure of the main window rather than by using child windows.

### Multiple Document Interface Windows

Windows MDI provides applications with a standard interface for displaying multiple documents within the same instance of an application. An MDI application creates a frame window that contains a client window in place of its client area. An application creates an MDI client window by calling CreateWindow with the class MDICLIENT and passing a CLIENTCREATESTRUCT structure as the function's lpParam parameter. This client window in turn can own multiple child windows, each of which displays a separate document. An MDI application controls these child windows by sending messages to its client window.

### Title Bar

The title bar, a rectangle at the top of the window, provides space for the window title or name. An application defines the window title when it creates the window. It can also change this name anytime by using the SetWindowText function. A title bar makes it possible for the user to move the window by using a mouse or other pointing device.

### System Menu

The System menu, identified by a box at the left end of the title bar, is a pop-up menu that contains the system commands. (The System menu is sometimes referred to as the Control menu.) The system commands are commands that can be selected by the user to direct Windows to carry out actions that affect the window, such as moving and closing it.

To create a window with a System menu or Close box, the application must specify both the WS\_SYSMENU and WS\_CAPTION window styles when the window is created.

### Scroll Bars

The horizontal and vertical scroll bars are bars on the lower and right sides of a window, respectively, making it possible for a user to scroll the contents of the client area. Windows sends scroll requests to a window as WM\_HSCROLL and WM\_VSCROLL messages. If the window permits scrolling, the window procedure must process these messages.

A window can have one or both scroll bars. To create a window with a scroll bar, the application must specify the WS\_HSCROLL or WS\_VSCROLL window style when the window is created. An application can use the ShowScrollBar function to show or hide a scroll bar of a window with the WS\_HSCROLL or WS\_VSCROLL style.

### Menus

A menu is a list of commands from which the user can select using the mouse or other pointing device or the keyboard. When the user selects an item, Windows sends a corresponding message to the window procedure to indicate which command was selected. Windows provides two types of menus: menu bars (sometimes called static menus) and pop-up menus.

A menu bar is a horizontal menu that appears at the top of a window and below the title bar, if one exists. Any window except a child window can have a menu bar. If an application does not specify a menu when it creates a window, the window receives the default menu bar (if any) defined by the window class.

A pop-up menu contains a vertical list of items and is often displayed when a user selects a menu-bar item. In turn, a pop-up menu item can display another pop-up menu. A pop-up menu can float--that is, it can appear anywhere on the screen designated by the application. An application creates an empty pop-up menu by calling the CreatePopupMenu function, and then fills in the menu using the AppendMenu and InsertMenu functions. It displays the pop-up menu by calling TrackPopupMenu.

An application can create or modify an individual menu item with the MF\_OWNERDRAW style, indicating that the item is an owner-drawn item. In this case, the owner of the menu is responsible for drawing all visual aspects of the menu item, including checked, grayed, and highlighted states. When the menu is displayed for the first time, the window that owns the menu receives a WM\_MEASUREITEM message. The lParam parameter of this message points to a MEASUREITEMSTRUCT structure. The owner then fills in this structure with the dimensions of the item and returns. Windows uses the information in the structure to determine the size of the item so that Windows can appropriately detect the user's interaction with the item. Windows sends the WM\_DRAWITEM message whenever the owner of the menu must update the visual appearance of an owner-drawn menu item. A top-level menu item cannot be an owner-drawn item.

An application can call the AppendMenu, InsertMenu, or ModifyMenu function to add an owner-drawn menu item to a menu or to change an existing menu item to be an owner-drawn menu item. To maintain additional data associated with the item, the application can supply a 32-bit value for the lpNewItem parameter of the function. This value is available to the application as the itemData member of the structures pointed to by the lParam parameter of the WM\_MEASUREITEM and WM\_DRAWITEM messages. For example, if an application were to draw the text in a menu item by using a specific color, the 32-bit value could contain a pointer to a string. The application could then set the text color before drawing the item when it received the WM\_DRAWITEM message.

#### Window State

The window state can be open (minimized, maximized, or restored), hidden or visible, and enabled or disabled. The initial state of a window depends on whether the following window styles are used:

- WS\_DISABLED
- WS\_MINIMIZE
- WS\_MAXIMIZE
- WS\_VISIBLE

By default, Windows creates windows that are initially enabled--that is, windows that can start receiving input messages immediately. An application can disable input to a new window by specifying the WS\_DISABLED window style.

A new window is not displayed until an application opens it by using the ShowWindow function or specifies the WS\_VISIBLE window style when it creates the window. For overlapped windows, the WS\_ICONIC window style creates a window that is minimized initially.

#### Life Cycle of a Window

Because the purpose of any window is to make it possible for the user to specify data or for the application to display information, a window starts its life cycle when the application has a need for input or output. A window continues its life cycle until there is no longer a need for it or the application is closed. Some windows, such as the window used for the application's main user interface, last the life of the application. Other windows, such as a window used as a dialog box, may last only a few seconds.

The first step in a window's life cycle is creation. Given a registered window class with a corresponding window procedure, the application uses the CreateWindow function to create the window. This function directs Windows to prepare internal structures for the window and to return a unique integer value, called a window handle, that the application can use to identify the window in subsequent function calls.

The first message most windows process is WM\_CREATE, the window-creation message. The CreateWindow function sends this message to inform the window procedure that it can now perform any initialization, such as allocating memory and preparing data files. The wParam parameter is not used, but the lParam parameter contains a long pointer to a CREATESTRUCT structure, whose members correspond to the parameters passed to CreateWindow.

The WM\_CREATE message is sent directly to the window procedure, bypassing the application's message queue. This means an application creates a window and processes the WM\_CREATE message before it enters the main message loop.

After a window has been created, it must be opened (displayed) before it can be used. An application can open the window in one of two ways: It can specify the WS\_VISIBLE window style in the CreateWindow function to open the window immediately after creation, or it can wait until later and call the ShowWindow function to open the window. When creating a main window, an application should not specify WS\_VISIBLE, but should call ShowWindow from the WinMain function with the nCmdShow parameter set to specify the window state.

When the window is no longer needed or the application is terminated, the window must be destroyed. This is done by using the DestroyWindow function. DestroyWindow removes the window from the system display and invalidates the window handle. It also sends WM\_DESTROY and

WM\_NCDESTROY messages to the window procedure. The DestroyWindow function also destroys all of the window's child and owned windows.

The window procedure also receives a WM\_DESTROY message when the WM\_CLOSE message is processed by the DefWindowProc function. When a window procedure receives a WM\_DESTROY message, it should free any allocated memory and close any open data files.

The window used as the application's main user interface should always be the last window destroyed and should always cause the application to terminate. When this window receives a WM\_DESTROY message, it should call the PostQuitMessage function. This function copies a WM\_QUIT message to the application's message queue as a signal for the application to close when the message is read from the queue.

## Painting

This section describes the system display and the preparation of windows for painting and other general-purpose graphics operations.

### How Windows Manages the Display

The system display is the principal display device for all applications running with Windows. All applications are free to display some form of output on the system display; but because many applications can run at one time, the complete system display must be shared. Windows shares the system display by carefully managing the access that applications have to it. Windows ensures that each application has space to display output but does not draw in the space reserved for other applications.

Windows manages the system display by using display contexts. The display context is a special device context that treats each window as a separate display surface. An application that retrieves a display context for a specific window has complete control of the system display within that window, but cannot access or paint over any part of the display outside the window.

### Display Context Types

There are four types of display contexts: common, class, private, and window. The common, class, and private display contexts permit drawing in the client area of a given window. The window display context permits drawing anywhere in the window. When a window is created, Windows assigns a common, class, or private display context to it, based on the type of display context specified in that window's class style. A window display context can be used for painting within a window's nonclient area.

### Common Display Context

A common display context is the default context for all windows. Windows assigns a common display context to the window if a display-context type is not explicitly specified in the window's class style.

A common display context permits drawing in a window's client area, but it is not immediately available for use by a window. A common display context must be retrieved from a cache of display contexts before a window can carry out any drawing in its client area. The `GetDC` or `BeginPaint` function retrieves the display context and returns a handle of the context. The handle can be used with GDI functions to draw in the client area of the given window. After drawing is complete, an application must use the `ReleaseDC` or `EndPaint` function to return the context to the cache. After the context is released, drawing cannot occur until another display context is retrieved.

When a common display context is retrieved, Windows gives it default selections for the tools currently available to carry out the actual drawing. The default selections for a common display context are as follows:

Attribute	Default
Background color	Background color setting from Windows Control Panel (typically, white).
Background mode	OPAQUE.
Bitmap	No default.
Brush	WHITE_BRUSH.
Brush origin	(0,0).
Clipping region	Entire client area with the update region clipped as appropriate. Child and pop-up windows in the client area may also be clipped.
Color palette	DEFAULT_PALETTE.
Current pen position	(0,0).
Device origin	Upper-left corner of client area.
Drawing mode	R2_COPYPEN.
Font	SYSTEM_FONT (SYSTEM_FIXED_FONT for applications written to run with Windows versions 3.0 or earlier).
Intercharacter spacing	0.
Mapping mode	MM_TEXT.
Pen	BLACK_PEN.
Polygon-filling mode	ALTERNATE.
Relative-absolute flag	ABSOLUTE.
Stretching mode	BLACKONWHITE.

Text color	Text color setting from Control Panel (typically, black).
Viewport extent	(1,1).
Viewport origin	(0,0).
Window extent	(1,1).
Window origin	(0,0).

An application can modify the attributes of the display context by using the selection functions and display-context attribute functions. For example, applications typically change the selected pen, brush, and font.

When a common display context is released, the current selections, such as mapping mode and clipping region, are lost. Windows does not preserve the previous selections of a common display context. Applications that modify the attributes of a common display context must do so each time another context is retrieved.

### Class Display Context

A window has a class display context if the window class specifies the `CS_CLASSDC` style. A class display context is shared by all windows in a given class. A class display context is not part of the display context cache. Instead, Windows specifically allocates a class context for exclusive use by the window class.

A class display context must be retrieved before it can be used, but it does not have to be released after use. As long as only one window from the class uses the context, the class display context can be kept and reused. If another window in the class needs to use the context, that window must retrieve it before any drawing occurs. Retrieving the context sets the correct device origin and clipping region for the new window and ensures that the context is applied to the correct window. An application can use the `GetDC` or `BeginPaint` function to retrieve a handle of the class display context. The `ReleaseDC` and `EndPaint` functions have no effect on a class display context.

A class display context is given the same default selections as a common display context when the first window of the class is created. These selections can be modified at any time. Windows preserves all new selections made for the class display context, except for the clipping region and device origin, which are adjusted for the current window when the context is retrieved. This means a change made by one window applies to all windows that subsequently use the context.

Note: Changing the mapping mode of a class display context may have an undesirable effect on how a window's background is erased. For more information, see [Window Background](#), and [Graphics Device Interface Overview](#).

### Private Display Context

A window has a private display context if the window class specifies the `CS_OWNDC` style. A private display context is used exclusively by a given window. A private display context is not part of the display context cache. Instead, Windows specifically allocates the context for exclusive use by the window. Although using private display contexts is convenient, they are expensive in terms of system resources, so an application should use them sparingly.

A private display context needs to be retrieved only once. Thereafter, it can be kept and used any number of times by the window. Windows automatically updates the context to reflect changes to the window, such as moving or sizing. An application can use the `GetDC` or `BeginPaint` function to retrieve a handle of a private display context. The `ReleaseDC` and `EndPaint` functions have no effect on a private display context.

A private display context is given the same default selections as a common display context when the window is created. These selections can be modified at any time. Windows preserves any new selections made for the context. New selections, such as of a clipping region or brush, remain selected until the window specifically makes a change.

Note: Changing the mapping mode of a private display context may have an undesirable effect on how the window's background is erased. For more information, see [Window Background](#), and [Graphics Device Interface Overview](#).

### Window Display Context

A window display context permits painting anywhere in a window, including the title bar, menus, and scroll bars. Its origin is the upper-left corner of the window instead of the upper-left corner of the client area.

The GetWindowDC function retrieves a window display context from the same cache as it does common display contexts. Therefore, a window that uses a window display context must release it with the ReleaseDC function immediately after drawing.

Windows always sets the current selections of a window display context to the same default selections as a common display context and does not preserve any change the window may have made to these selections. The CS\_OWNDC and CS\_CLASSDC class styles have no effect on the window display context.

A window display context is intended to be used for special painting within a window's nonclient area. Because painting in nonclient areas of overlapped windows is not recommended, most applications reserve a display context for designing custom child windows. For example, an application can use the display context to draw a custom border around the window. In such cases, the window usually processes the WM\_NCPAINT message instead of passing it to the DefWindowProc function. For applications that do not process WM\_NCPAINT messages but still need to paint within the nonclient area, the GetSystemMetrics function can be used to retrieve the dimensions of various parts of the nonclient area, such as the title bar, menu bar, and scroll bars.

### Display-Context Cache

Windows maintains a cache of display contexts that it uses for common display contexts and window display contexts. This cache contains five display contexts, which means only five common display contexts can be active at any one time. To prevent more than five from being retrieved, a window that uses a common or window display context must release that context immediately after drawing.

If a window fails to release a common display context, all five display contexts may eventually be active and unavailable for any other window. In such a case, Windows ignores all subsequent requests for a common display context. In the retail version of Windows, the system appears to be deadlocked, while the debugging version of Windows undergoes a fatal exit, alerting you of a problem.

The ReleaseDC function releases a display context and returns it to the cache. Class and private display contexts are individually allocated for each class or window; they do not belong to the cache, so they do not need to be released after use.

### Painting Sequence

To manage the system display, Windows carries out many operations that affect the contents of the client area. If Windows moves, sizes, or alters the appearance of the screen, the change may affect a given window. If so, Windows marks the area changed by the operation as ready for updating and, at the next opportunity, sends a WM\_PAINT message to the window so that it can update the window in the update region. If a window paints in its client area, it must call the BeginPaint function to retrieve a handle of a display context, must update the changed area as defined by the update region, and finally, must call the EndPaint function to complete the operation.

A window can paint within its client area at any time--that is, at times other than in response to a WM\_PAINT message. The only requirement is that it retrieve a display context for the client area before carrying out any operations.

### WM\_PAINT Message

The WM\_PAINT message is a request from Windows to a given window to update its display. Windows sends a WM\_PAINT message to a window whenever it is necessary to repaint a portion of the window. When a window receives a WM\_PAINT message, it should retrieve the update region by using the BeginPaint function, and it should carry out whatever operations are necessary to update that part of the client area.

The InvalidateRect and InvalidateRgn functions do not actually generate WM\_PAINT messages. Instead, Windows accumulates the changes made by these functions and its own changes while a window processes other messages in its message queue. Postponing the WM\_PAINT message lets a window process all changes at once instead of updating bits and pieces in time-consuming individual steps.

To direct Windows to send a WM\_PAINT message, an application can use the UpdateWindow function. The UpdateWindow function sends the message directly to the window, regardless of the number of other messages in the application's message queue. UpdateWindow is typically used when a window needs to update its client area immediately, such as just after the window is created.

Once a window receives a WM\_PAINT message, it must call the BeginPaint function to retrieve the display context for the client area and to retrieve other information such as the update region and whether the background has been erased.



Windows automatically selects the update region as the clipping region of the display context. Since GDI discards (clips) drawing that extends outside the clipping region, only drawing that is in the update region is actually visible. For more information about the clipping region, see Graphics Device Interface Overview.

The BeginPaint function clears the update region to prevent the same region from generating subsequent WM\_PAINT messages.

After completing the painting operation, the window must call the EndPaint function to release the display context.

### Update Region

An update region defines the part of the client area that is marked for painting on the next WM\_PAINT message. The purpose of the update region is to save applications the time it takes to paint the entire contents of the client area. If only the part that needs painting is added to the update region, only that part is painted. For example, if a word changes in the client area of a word-processing application, only the word needs to be painted, not the entire line of text. This saves the time it takes the application to draw the text, especially if there are many different sizes and fonts.

The InvalidateRect and InvalidateRgn functions add a given rectangle or region to the update region. The rectangle or region must be given in client coordinates. The update region itself is defined in client coordinates. Windows adds its own rectangles and regions to a window's update region after operations such as moving, sizing, and scrolling the window.

The ValidateRect and ValidateRgn functions remove a given rectangle or region from the update region. These functions are typically used when the window has updated a specific part of the display in the update region before receiving the WM\_PAINT message.

The GetUpdateRect function retrieves the smallest rectangle that encloses the entire update region. The GetUpdateRgn function retrieves the update region itself. These functions can be used to compute the current size of the update region to determine if painting is required.

### Window Background

The window background is the color or pattern the client area is filled with before a window begins painting in the client area. Windows paints the background for a window or gives the window the opportunity to do so by sending a WM\_ERASEBKGD message to the window when the application calls the BeginPaint function.

The background is important because if it is not erased, the client area will contain whatever was originally on the screen before the window was moved there. Windows erases the background by filling it with the background brush specified by the window's class.

Windows applications that use class or private display contexts should be careful about erasing the background. Windows assumes the background is to be computed by using the MM\_TEXT mapping mode. If the display context has any other mapping mode, the area erased may not be within the visible part of the client area.

### Brush Alignment

Brush alignment is particularly important on the system display where scrolling and moving are commonplace. A brush is a pattern of bits with a minimum size of 8-by-8 bits. GDI paints with a brush by repeating the pattern again and again within a given rectangle or region. If the region is moved by an arbitrary amount--for example, if the window is scrolled--and the brush is used again to fill empty areas around the original area, there is no guarantee that the original pattern and the new pattern will be aligned. For example, if the scroll moves the original filled area up one pixel, the intersection of the original area and any new painting will be out of alignment by one pixel, or bit. Depending on the pattern, this may have an undesirable visual effect. For more information about brushes, see Graphics Device Interface Overview.

To ensure that a brush is aligned after a window is moved, an application must take the following steps:

- 1 Call the SelectObject function to select a different brush to be the current brush.
- 2 Call the SetBrushOrg function to realign the current brush.
- 3 Call the UnrealizeObject function to realign the origin of the original brush when it is selected next. (UnrealizeObject should not be used on stock objects, only on brushes created by the application.)
- 4 Call the SelectObject function to select the original brush.

### Painting Rectangular Areas

The FillRect, FrameRect, and InvertRect functions provide an easy way to carry out painting operations on rectangles in the client area.

The FillRect function fills a rectangle with the color and pattern of a given brush. This function fills all parts of the rectangle, including the edges or borders.

The FrameRect function uses a brush to draw a border around a rectangle. The border width and height is one unit.

The InvertRect function inverts the contents of the given rectangle. On monochrome displays, white pixels become black, and vice versa. On color displays, the results depend on the method used by the display to generate color. In either case, calling InvertRect twice with the same rectangle restores the screen to its original colors.

#### Drawing Icons

The DrawIcon function draws an icon at a given location in the client area. An icon is a bitmap that a window uses as a symbol to represent an item, such as an application or a warning.

You can use the Image Editor to create an icon and then use Microsoft Windows Resource Compiler (RC) to add the icon to your application's resources. Your application can then call the LoadIcon function to load the icon into memory.

Applications can also call the CreateIcon function to create an icon and can modify a previously loaded or created icon at any time. An icon resource is in global memory, and the icon's handle is the handle of that memory. An application can free memory used to store an icon created by CreateIcon by calling the DeleteIcon function.

#### Drawing Formatted Text

The DrawText function formats and draws text within a given rectangle in the client area. This function provides simple text processing that most applications can use to display text. DrawText output is similar to the output generated by a terminal, except it uses the selected font and can clip the text if it extends outside a given rectangle. DrawText provides many different formatting styles.

The DrawText function uses the currently selected font, so applications can draw formatted text in a font other than the system font.

DrawText does not hyphenate, and although it can left align, right align, or center text, it cannot combine alignment styles. In other words, it cannot align to both the left and right.

DrawText recognizes a number of control characters and carries out special actions when it encounters them. The control characters and their respective actions are as follows:

Windows character	Action
Carriage return (13)	Interpreted as a line-break character. The text is immediately broken and continued on the next line down in the rectangle.
Linefeed (10)	Interpreted as a line-break character. The text is immediately broken and continued on the next line down in the rectangle. A carriage return–linefeed character combination is interpreted as a single line-break character.
Space (32)	Interpreted as a wordwrap character if the <u>DT_WORDBREAK</u> style is given. If the text is too long to fit on the current line in the formatting rectangle, the line is broken at the wordwrap character that is closest to the end of the line.
Tab (9)	Expanded into a given number of spaces if the <u>DT_EXPANDTABS</u> style is given. The number of spaces depends on which tab-stop value is given with the <u>DT_TABSTOP</u> style. The default value is eight.

#### Drawing Gray Text

An application can draw gray text by calling the SetTextColor function to set the current text color to COLOR\_GRAYTEXT, the solid gray system color used to draw disabled text. However, if the current display driver does not support a solid gray color, this value is set to zero.

The GrayString function is a multiple-purpose function that gives applications another way to gray text or carry out other customized operations on text or bitmaps before drawing the result in a client area. To gray text, the function creates a memory bitmap, draws the string in the bitmap, and then grays the string by combining it with a gray brush. The GrayString function finally copies the gray text to the display. However, an application can intercept or modify each step of this process to carry out custom effects, such as changing the gray brush to a patterned brush or drawing an icon instead of a string.

If GrayString is used to draw gray text only, GrayString uses the selected font of the given display context. First, GrayString sets text color to black. It then creates a bitmap and uses the TextOut function to write a given string to the bitmap. It then uses the PatBlt function and a gray brush to gray the text, and uses the BitBlt function to copy the bitmap to the client area.

GrayString assumes that the display context for the client area has MM\_TEXT mapping mode. Other mapping modes cause undesirable results.

GrayString lets an application modify this graying procedure in three ways: by defining an additional brush to be combined with the text before the text is displayed, by replacing the call to the TextOut function with a call to an application-supplied function, and by disabling the call to the PatBlt function.

If an additional brush is combined with text, it is defined for the hbr parameter of GrayString. The brush is combined with the text as the text is copied to the client area by the BitBlt function. The additional brush is intended to be used to give the text a desired color, because the bitmap used to draw the text is a monochrome bitmap.

If an application-supplied function replaces TextOut, it is defined for the gsprc parameter of GrayString. When gsprc is not NULL, GrayString automatically calls the application-supplied function instead of the TextOut function and passes it a handle of the display context for the memory bitmap and the long pointer and count passed to GrayString. The function can carry out any operation and interpret the long pointer and count in any way. For example, a negative count could be used to indicate that the long pointer points to an icon handle that signals the application-supplied function to draw the icon and let GrayString gray and display it. No matter what type of drawing the function carries out, GrayString assumes it is successful if the application-supplied function returns a nonzero value.

GrayString suppresses graying if it receives a cch parameter equal to -1 and the application-supplied function returns zero. This provides a way to combine custom patterns with the text without interference from the gray brush.

#### Nonclient-Area Painting

Windows sends a WM\_NCPAINT message to the window whenever a part of the nonclient area of the window, such as the title bar, menu bar, or window frame, needs painting. Processing this message is not recommended because a window that does so must be able to paint all the required parts of the nonclient area for the window. Unless the Windows application is creating a custom nonclient area for a child window, a window should pass this message to the DefWindowProc function for default processing.

## Dialog Boxes

A dialog box is a temporary window that Windows creates for special-purpose input and then destroys immediately after use. An application typically uses a dialog box to prompt the user for additional information about a current command selection.

### Uses for Dialog Boxes

For convenience and to keep from introducing device-dependent values into the application code, applications use dialog boxes instead of creating their own windows. This device independence is maintained by using logical coordinates in the dialog box template. A dialog box is convenient to use because all aspects of the dialog box, except how to carry out its tasks, are predefined. A dialog box supplies a window class and procedure; the window for the dialog box is created automatically. The application supplies a dialog box procedure to carry out tasks and a dialog box template that describes the dialog box style and content.

### Modeless Dialog Box

A modeless dialog box allows the user to supply information to the dialog box and return to the previous task without canceling or removing the dialog box. A modeless dialog box makes it possible for a user to supply more than one piece of information about the current task without having to select a command from a menu each time. For example, a modeless dialog box is often used with a text-search command in word-processing applications. The dialog box remains displayed while the search is carried out. The user can then return to the dialog box and search for the same word again, or change the entry in the dialog box and search for a new word.

An application with a modeless dialog box processes messages for that box by using the IsDialogMessage function inside the main message loop. The dialog box procedure of a modeless dialog box must send a message to the parent window when it has input for the parent window. The dialog box procedure must also destroy the dialog box when it is no longer needed. An application can call the DestroyWindow function to destroy a modeless dialog box. The application must not call the EndDialog function to destroy a modeless dialog box.

### Modal Dialog Box

A modal dialog box requires the user to respond to a request before the application continues. Typically, a modal dialog box is used when a chosen command needs additional information before it can proceed.

A modal dialog box disables its parent window, and it creates its own message loop, temporarily taking control of the application's message queue from the application's main message loop.

By default, a modal dialog box cannot be moved by the user. An application can create a movable modal dialog box by specifying the WS\_CAPTION window style.

The dialog box is displayed until the dialog box procedure calls the EndDialog function, or until Windows is closed. The parent window remains disabled unless the dialog box enables it. Note that enabling the parent window is not recommended because it defeats the purpose of the modal dialog box.

### System-Modal Dialog Box

A system-modal dialog box is identical to a modal dialog box except that all windows, not just the parent window, are disabled. System-modal dialog boxes must be used with care because they effectively shut down the system until the user supplies the required information.

### Creating a Dialog Box

A dialog box is typically created by using either the CreateDialog or DialogBox function. These functions load a dialog box template from the application's executable file and then create a pop-up window that matches the template's specifications. The dialog box belongs to the predefined dialog box class unless another class is explicitly defined. The DialogBox function creates a modal dialog box; the CreateDialog function creates a modeless dialog box.

Use the WS\_VISIBLE style for the dialog box template if you want the dialog box to appear upon creation.

### Dialog Box Template

The dialog box template is a description of the dialog box: its height and width, the controls it contains, its style, the type of border it uses, and so on. A template is an application's resource. You use the Resource Compiler to convert the text description of the template to the required binary form and to add that binary form to the application's executable file.

Because a dialog box is system-independent, you can easily modify the template without changing the source code.

The CreateDialog or DialogBox function loads the resource into memory when it creates the dialog box and then uses the information in the dialog box template to create the dialog box, position it, and create and position the controls for the dialog box.

#### Dialog Box Measurements

Dialog box and control dimensions and coordinates are device-independent. Because a dialog box may be displayed on system displays that have widely varying pixel resolutions, dialog box dimensions are specified in system-character widths and heights instead of pixels. This ensures the best possible appearance of characters. One unit in the x-direction is equal to one-fourth of the dialog box base width unit. One unit in the y-direction is equal to one-eighth of the dialog box base height unit. The dialog box base units are computed from the height and width of the system font; the GetDialogBaseUnits function returns the dialog box base units for the current display. Applications can convert these measurements to pixels by using the MapDialogRect function.

Windows does not allow the height of a dialog box to exceed the height of a full-screen window, and it does not allow the width of a dialog box to be greater than the width of the screen.

#### Return Values from a Dialog Box

The DialogBox function that creates a modal dialog box does not return until the dialog box procedure has called the EndDialog function to signal the destruction of the dialog box. When control finally returns from the DialogBox function, the return value is equal to the value specified in the EndDialog function. This means a modal dialog box can return a value through the EndDialog function.

Modeless dialog boxes cannot return values in this way because they do not use the EndDialog function to close and do not return control in the same way a modal dialog box does. Instead, a modeless dialog box returns values to its parent window by using the SendMessage function to send a notification message to the parent window. Although Windows does not explicitly define the content of a notification message, most applications use a WM\_COMMAND message with an integer value that identifies the dialog box in the wParam parameter and the return value in the lParam parameter. A modal dialog box can also use this technique to return values to its parent window before closing.

#### Controls in a Dialog Box

A control is a child window that belongs to a predefined or application-defined window class and that gives the user a method of supplying input to the application. A dialog box can contain any number and any types of controls. Examples of controls are push buttons and edit controls. Most dialog boxes contain one or more controls of the predefined class. The number of controls, the order in which they should be created, and the location of each in the dialog box are defined by the control statements given in the dialog box template.

#### Control Identifiers

Every control in a dialog box needs a unique control identifier, or ID, to distinguish it from other controls. Because all controls send information to the dialog box procedure through WM\_COMMAND messages, the control identifiers are essential for the dialog box to determine which control sent a given message.

Each control in the dialog box must have a unique identifier. If a dialog box has a menu bar, there must be no conflict between menu-item identifiers and control identifiers. Because Windows sends menu input to a dialog box procedure as WM\_COMMAND messages, conflicts with menu and control identifiers can cause errors. Menus in dialog boxes are not recommended.

The dialog box procedure usually identifies each dialog box control by using its control identifier. Occasionally the dialog box procedure requires the window handle that was given to the control when it was created. The dialog box procedure can retrieve this window handle by using the GetDlgItem function.

#### The WS\_TABSTOP and WS\_GROUP Control Styles

The WS\_TABSTOP style specifies that the user can move the input focus to the given control by pressing the TAB key or SHIFT+TAB keys. Typically, every control in the dialog box has this style, so the user can move the input focus from one control to the other. If two or more controls are in the dialog box, the TAB key moves the input focus to the controls in the order in which they have been created. The SHIFT+TAB keys move the input focus in reverse order. For modal dialog boxes, the TAB and SHIFT+TAB keys are automatically enabled for moving the input focus. For modeless dialog boxes, the

IsDialogMessage function must be used to filter messages for the dialog box and to process these keystrokes. Otherwise, the keys have no special meaning and the `WS_TABSTOP` style is ignored.

The `WS_GROUP` style specifies that the user can move the input focus within a group of controls by using the arrow keys. The first control in a group of controls must have the `WS_GROUP` style. The next control that has the `WS_GROUP` style marks the bottom boundary of the group; the input focus cannot be moved to this control by using the arrow keys. The `DOWN ARROW` and `RIGHT ARROW` keys move the input focus to controls in the order in which they have been created. The `UP ARROW` and `LEFT ARROW` keys move the input focus in reverse order. For modal dialog boxes, the arrow keys are automatically enabled for moving the input focus. For modeless dialog boxes, the `IsDialogMessage` function must be used to filter messages for the dialog box and to process these keystrokes. Otherwise, the keys have no special meaning and the `WS_GROUP` style is ignored.

## Buttons

Buttons are the principal interface of a dialog box. Almost all dialog boxes have at least one push button, and most have one default push button (a push button having the `BS_DEFPUSHBUTTON` style) and one or more other push buttons. Many dialog boxes have collections of radio buttons enclosed in group boxes or have lists of check boxes.

Most modal or modeless dialog boxes that use the special keyboard interface have a default push button whose control identifier is set to `IDOK` so that the action the dialog box procedure takes when the button is chosen is identical to the action taken when the `ENTER` key is pressed. There can be only one button with the default style; however, an application can assign the default style to any button at any time. Most dialog boxes that use the special keyboard interface can also set the control identifier of another push button to `IDCANCEL` so that the action of the `ESC` key is duplicated by choosing that button.

When a dialog box first starts, the dialog box procedure can set the initial state of each button by using the `CheckDlgButton` function, which sets or clears the button state. This function is most useful when used to set the state of radio buttons or check boxes. If the dialog box contains a group of radio buttons in which only one button should be set at any given time, the dialog box procedure can use the `CheckRadioButton` function to set the appropriate radio button and automatically clear any other radio button.

Before a dialog box terminates, the dialog box procedure can check the state of each button control by using the `IsDlgButtonChecked` function, which returns the current state of the button. A dialog box typically saves this information to initialize the buttons the next time the dialog box is created.

## Edit Controls

Many dialog boxes have edit controls that let the user supply text as input. Most dialog box procedures initialize an edit control when the dialog box first starts. For example, the dialog box procedure may place a proposed filename in the control that the user can select, modify, or replace. The dialog box procedure can set the text in an edit control by using the `SetDlgItemText` function, which copies text from a given buffer to the edit control. When the edit control receives the input focus, the complete text is automatically selected for editing.

Because edit controls do not automatically return their text to the dialog box, the dialog box procedure must retrieve the text before terminating. It can retrieve the text by using the `GetDlgItemText` function, which copies the edit-control text to a buffer. The dialog box procedure typically saves this text to initialize the edit control later or passes it on to the parent window for processing.

Some dialog boxes use edit controls that let the user enter numbers. The dialog box procedure can retrieve a number from an edit control by using the `GetDlgItemInt` function, which retrieves the text from the edit control and converts the text to a decimal value. The user enters the number in decimal digits. It can be either signed or unsigned. The dialog box procedure can display an integer by using the `SetDlgItemInt` function. `SetDlgItemInt` converts a signed or unsigned integer to a string of decimal digits.

## List Boxes and Directory Listings

Some dialog boxes display lists, such as a list of filenames, from which the user can select one or more items. To display a list of filenames, a dialog box typically uses a list box and the `DlgDirList` and `DlgDirSelect` functions. The `DlgDirList` function automatically fills a list box with the filenames in the current directory. The `DlgDirSelect` function retrieves the selected filename from the list box. Together, these two functions provide a convenient way for a dialog box to display a directory listing that makes it possible for the user to select a file without having to type the location and name of the file.

## Combo Boxes

Another method for providing a list of items to a user is by using a combo box. A combo box consists of either a static control or edit control combined with a list box. The list box can be displayed at all times or pulled down by the user. If the combo box contains a static control, that control always displays the current selection (if any) from the list box portion of the combo box. If the combo box uses an edit control, the user can type a selection; the list box highlights the first item (if any) that matches what the user has entered in the edit control. The user can choose the OK button or press ENTER to complete the choice.

#### Owner-Drawn Dialog Box Controls

List boxes, combo boxes, and buttons can be designated as owner-drawn controls by creating them with the appropriate style. Following are available styles:

Style	Meaning
LBS_OWNERDRAWFIXED	Creates an owner-drawn list box with items that have the same, fixed height.
LBS_OWNERDRAWVARIABLE	Creates an owner-drawn list box with items that have different heights.
CBS_OWNERDRAWFIXED	Creates an owner-drawn combo box with items that have the same, fixed height.
CBS_OWNERDRAWVARIABLE	Creates an owner-drawn combo box with items that have different heights.
BS_OWNERDRAW	Creates an owner-drawn button.

When a control has the owner-drawn style, Windows handles the user's interaction with the control as usual, performing such tasks as detecting when a user has chosen a button and notifying the button's owner of the event. However, because the control is owner-drawn, the owner of the control is completely responsible for the visual appearance of the control. Owner-drawn list boxes and combo boxes can control the display of only the individual elements within a list box or combo box, not the entire list box or combo box.

When Windows first creates a dialog box containing owner-drawn controls, it sends the owner a WM\_MEASUREITEM message for each owner-drawn control. The lParam parameter of this message contains a pointer to a MEASUREITEMSTRUCT structure. When the owner receives the message for a control, the owner fills in the appropriate members of the structure and returns. This informs Windows of the dimensions of the control or of its items so that Windows can appropriately detect the user's interaction with the control. If a list box or combo box is created with the LBS\_OWNERDRAWVARIABLE or CBS\_OWNERDRAWVARIABLE style, the WM\_MEASUREITEM message is sent to the owner for each item in the control, because each item can differ in height. Otherwise, this message is sent once for the entire owner-drawn control.

Whenever an owner-drawn control needs to be redrawn, Windows sends the WM\_DRAWITEM message to the owner of the control. The lParam parameter of this message contains a pointer to a DRAWITEMSTRUCT structure that contains information about the drawing required for the control. Similarly, if an item is deleted from a list box or combo box, Windows sends the WM\_DELETEITEM message containing a pointer to a DELETEITEMSTRUCT structure that describes the deleted item.

#### Messages for Dialog Box Controls

Many controls recognize predefined messages that, when sent to the control, cause it to carry out some action. A dialog box procedure can send a message to a control by supplying the control identifier and using the SendDlgItemMessage function, which is identical to the SendMessage function except that it uses a control identifier instead of a window handle to identify the control that is to receive the message.

#### Keyboard Interface for Dialog Boxes

Windows provides a special keyboard interface for modal dialog boxes and modeless dialog boxes that use the IsDialogMessage function to filter messages. This keyboard interface carries out special processing for several keys and generates messages that correspond to certain buttons in the dialog box or change the input focus from one control to another. The keys used in this interface and the respective actions are as follows:

Key	Action
DOWN ARROW	Moves the input focus to the next control in the group.
ENTER	Sends a <u>WM_COMMAND</u> message to the dialog box procedure. The wParam parameter is set to 1 or the default button.

ESC	Sends a <u>WM_COMMAND</u> message to the dialog box procedure. The wParam parameter is set to 2.
LEFT ARROW	Moves the input focus to the previous control in the group.
RIGHT ARROW	Moves the input focus to the next control in the group.
SHIFT+TAB	Moves the input focus to the previous control that has the <u>WS_TABSTOP</u> style.
TAB	Moves the input focus to the next control that has the <u>WS_TABSTOP</u> style.
UP ARROW	Moves the input focus to the previous control in the group.

The TAB key and the arrow keys have no effect if the controls in the dialog box do not have the WS\_TABSTOP or WS\_GROUP style. The keys have no effect in a modeless dialog box if the IsDialogMessage function is not used to filter messages for the dialog box.

Note: For applications that use accelerator keys and have modeless dialog boxes, the IsDialogMessage function must be called before the TranslateAccelerator function. Otherwise, the keyboard interface for the dialog box may not be processed correctly.

Applications that have modeless dialog boxes and need those boxes to have the special keyboard interface must filter all messages retrieved from the application's message queue through the IsDialogMessage function before carrying out any other processing. This means that the application must pass the message to IsDialogMessage immediately after retrieving the message by using the GetMessage or PeekMessage function. Most applications that have modeless dialog boxes incorporate the IsDialogMessage function as part of the main message loop in the WinMain function. The IsDialogMessage function automatically processes any messages for the dialog box. This means that if the function returns a nonzero value, the message does not require additional processing and must not be passed to the TranslateMessage or DispatchMessage function.

The IsDialogMessage function also processes ALT+application-defined mnemonic key sequences.

In modal dialog boxes, the arrow keys have specific functions that depend on the controls in the box. For example, the keys move the input focus from control to control in group boxes, move the cursor in edit controls, and scroll the contents of list boxes. The arrow keys cannot be used to scroll the contents of any dialog box that has its own scroll bars. If a dialog box has scroll bars, the application must provide an appropriate keyboard interface for the scroll bars. Note that the mouse interface for scrolling is available if the system has a mouse.



## Scrolling

Scrolling is the movement of data in and out of the client area at the request of the user. It is a way for the user to see a document or graphic in parts if Windows cannot fit the entire document or graphic inside the client area. A scroll bar allows the user to control scrolling.

### Standard Scroll Bars and Scroll-Bar Controls

A standard scroll bar is a part of the nonclient area of a window. It is created with the window and displayed when the window is displayed. The sole purpose of a standard scroll bar is to let users generate scrolling requests for the window's client area. A window has standard scroll bars if it is created with the `WS_VSCROLL` or `WS_HSCROLL` style. A standard scroll bar is either vertical or horizontal. A vertical scroll bar, if used, always appears at the right of the client area; a horizontal scroll bar, if used, always appears at the bottom. A standard scroll bar always has the standard scroll-bar height and width as defined by the `SM_CXVSCROLL` and `SM_CYHSCROLL` system metric values.

A scroll-bar control is a control window that looks and acts like a standard scroll bar. But unlike a standard scroll bar, a scroll-bar control is not part of any window. As a separate window, a scroll-bar control can receive the input focus and indicates that it has the focus by displaying a flashing caret in the scroll box (also called the thumb). When a scroll-bar control has the input focus, the user can use the keyboard to direct the scrolling. Unlike standard scroll bars, a scroll-bar control provides a built-in keyboard interface. Scroll-bar controls also can be used for other purposes. For example, a scroll-bar control can be used to select values from a range of values, such as a color from a spectrum of colors.

### Scroll Box

The scroll box is the small rectangle in a scroll bar. It shows the approximate location within the current document or file of the data currently displayed in the client area. For example, the scroll box is in the middle of the scroll bar when page three of a five-page document is in the client area.

The `SetScrollPos` function sets the scroll box position in a scroll bar. Because Windows does not automatically update the scroll box position when an application scrolls, `SetScrollPos` must be used to update the position. The `GetScrollPos` function retrieves the current position.

A scroll box position is represented as an integer. The position is relative to the left or upper end of the scroll bar, depending on whether the scroll bar is horizontal or vertical. The position must be within the scroll-bar range, which is defined by minimum and maximum values. The positions are distributed equally along the scroll bar. For example, if the range is 0 through 100, there are 101 positions along the scroll bar, each equally spaced so that position 50 is in the middle of the scroll bar. The initial range depends on the scroll bar. Standard scroll bars have an initial range of 0 through 100; scroll-bar controls have an empty range (both minimum and maximum values are 0) if no explicit range is given when the control is created. An application can change the range by using the `SetScrollRange` function to set new minimum and maximum values so that applications can change the range at any time. The `GetScrollRange` function retrieves the current minimum and maximum values. The minimum and maximum values can be any integers. For example, a spreadsheet program with 255 rows can set the vertical scroll range to 1 through 255.

If `SetScrollPos` specifies a position value that is less than the minimum or more than the maximum, the minimum or maximum value is used instead. `SetScrollPos` moves the scroll box along the scroll bar.

### Scrolling Requests

A user makes a scrolling request by clicking in a scroll bar. Windows sends the request to the given window in the form of `WM_HSCROLL` and `WM_VSCROLL` messages. The messages' `lParam` parameter contains a position value and the handle of the scroll-bar control that generated the message (`lParam` is zero if a standard scroll bar generated the message). The `wParam` parameter specifies the type of scrolling; for example, the user may scroll up one line, scroll down a page, or scroll to the bottom. The type of scrolling is determined by which area of the scroll bar the user clicks.

The user can also make a scrolling request by using the scroll box, the small rectangle inside the scroll bar. The user moves the scroll box by moving the mouse while holding the left mouse button down when the cursor is positioned on the scroll box. The scroll bar sends `SB_THUMBTRACK` and `SB_THUMBPOSITION` flags with a `WM_HSCROLL` or `WM_VSCROLL` message to an application as the user moves the scroll box. Each message specifies the current position of the scroll box.

### Processing Scroll Messages

A window that permits scrolling needs a standard scroll bar or a scroll-bar control to let the user generate scrolling requests, and it needs a window procedure to process the `WM_HSCROLL` and `WM_VSCROLL` messages that represent the scrolling requests. Although the result of a scrolling

request depends entirely on how the window processes it, a window typically carries out a scroll operation by moving through the application's displayed information in some direction from the current location or to a known beginning or end and by displaying the data at the new location. For example, a word-processing application can scroll to the next line, the next page, or to the end of the document.

#### Scrolling the Client Area

The simplest way to scroll is to erase the current contents of the client area, and then paint the new information. This is the method an application is likely to use with `SB_PAGEUP`, `SB_PAGEDOWN`, `SB_TOP`, and `SB_END` requests, which require completely new contents.

For some requests, such as `SB_LINEUP` and `SB_LINEDOWN`, not all the contents need to be erased, since some are still visible after the scroll. The `ScrollWindow` function preserves a portion of the client area's contents, moves the preserved portion the specified amount, and prepares the rest of the client area for painting new information. `ScrollWindow` uses the `BitBlt` function to move a specific part of the client area to a new location within the client area. Any part of the client area that is uncovered (not in the part to be preserved) is invalidated and is erased and painted over at the next `WM_PAINT` message.

`ScrollWindow` also lets an application clip a part of the client area from the scroll. This keeps items that have fixed positions in the client area, such as child windows, from moving. This action automatically invalidates the part of the client area that is to receive the new information so that the application does not have to compute its own clipping regions.

#### Hiding a Standard Scroll Bar

For standard scroll bars, if the minimum and maximum values are equal, the scroll bar is hidden and, in effect, disabled. Using this technique, you can temporarily hide a scroll bar when it is not needed for the current contents of the client area.

The `SetScrollRange` function hides and disables a standard scroll bar when equal minimum and maximum values are specified. No scrolling requests can be made through the scroll bar when it is hidden. `SetScrollRange` enables the scroll bar and shows it again when it sets the minimum and maximum values to unequal values. The `ShowScrollBar` function can also be used to hide or show a scroll bar. It does not affect the scroll bar's range or scroll box's position.

## The Caret

The Windows caret is a flashing line, block, or bitmap that marks a location in a window's client area. The caret is especially useful in word-processing applications to mark a location in text for keyboard editing.

### Creating and Displaying a Caret

Windows forms a caret by inverting the pixel color within the rectangle given by the caret's position, width, and height. Windows flashes the caret by alternately inverting the display and restoring it to its previous appearance. The caret's flash rate, in milliseconds, defines the elapsed time between inverting and restoring the display. A complete flash (on-off-on) takes twice the blink time.

The CreateCaret function creates the caret shape and assigns ownership of the caret to the given window. The caret can vary in color and shape; a bitmap caret can be given any pattern.

Windows displays a solid caret by inverting everything in the rectangle defined by the caret's width and height. For a gray caret, Windows inverts every other pixel. For a pattern, Windows inverts only the white bits of the bitmap that defines the pattern. The width and height of a caret are given in logical units, which means they are subject to the window's mapping mode.

### Sharing the Caret

There is only one caret, so only one caret shape can be active at a time. All applications must cooperatively share the caret. Because Windows does not inform an application when a caret is created or destroyed, each window should create, move, show, or hide a caret only when it has the input focus or is active. A window should destroy the caret before losing the input focus or becoming inactive.

Your application can use the CreateBitmap function to create a bitmap for the caret; or, after you have used the Image Editor to create a bitmap and have used the Resource Compiler to add it to your application's resources, your application can use the LoadBitmap function to load the bitmap from the application's resources.

## The Cursor

The cursor is a bitmap, displayed on the screen. The user can use a mouse or other pointing device to move this bitmap to an item on the screen, such as a window or an icon. (In the remainder of section, the term mouse is used for any pointing device.)

### The Mouse and the Cursor

When a system has a mouse, the cursor shows the current location of the mouse. Windows automatically displays and moves the cursor when the mouse is moved. If a system does not have a mouse, Windows does not automatically display or move the cursor. Applications can use the cursor functions to display or move the cursor when a system does not have a mouse.

### Displaying and Hiding the Cursor

In a system without a mouse, Windows does not display or move the cursor unless the user chooses certain system commands, such as commands for sizing and moving. This means that after a call to the SetCursor function, the cursor remains on the screen until a subsequent call to SetCursor with the parameter set to NULL removes the cursor, or until a system command is carried out. Applications that need to use the cursor without a mouse usually simulate mouse input by using keys, such as the arrow keys, and display and move the cursor by using the cursor functions.

The ShowCursor function shows or hides the cursor. It is used to temporarily hide the cursor, and then restore it without changing the current cursor shape. This function actually sets an internal counter that determines whether the cursor should be drawn. Showing the cursor increments the counter; hiding the cursor decrements the counter. The cursor is only visible when the count is not a negative value.

### Positioning the Cursor

The SetCursorPos and GetCursorPos functions set and retrieve the current screen coordinates of the cursor. Although the cursor can be set at a location other than the current mouse location, if the system has a mouse any mouse movement causes the cursor to be redrawn at the mouse location. The SetCursorPos and GetCursorPos functions are most often used in applications that use the keyboard and specified keystrokes to move the cursor. Note that screen coordinates are not affected by the mapping mode in a window's client area.

### The Cursor Hot Spot and Confining the Cursor

The hot spot of the cursor is the location in the cursor bitmap that is tracked and recognized as the position of the mouse or keyboard arrow key. For example, the hot spot on the pointer is the point at the tip of the arrow.

The ClipCursor function confines the cursor to a given rectangle on the screen. The cursor can move to the edge of the rectangle but cannot move out of it. ClipCursor is typically used to restrict the cursor to a given window, such as a dialog box that contains a warning about a serious error. The rectangle is always given in screen coordinates and does not have to be within the window of the active application.

### Creating a Custom Cursor

The SetCursor function sets the cursor shape and draws the cursor. When a system has a mouse, Windows automatically changes the shape of the cursor when it crosses a window border or enters a different part of a window, such as a title or menu bar. Windows uses standard cursor shapes for the different parts of the screen, such as a pointer in a title bar. The SetCursor function lets an application delete the standard cursor and draw its own custom cursor. The cursor keeps its new shape until the mouse moves or a system command is carried out.

### Cursor Functions

Cursor functions set, move, show, hide, and confine the cursor. Following are the cursor functions:

Function	Description
<u>ClipCursor</u>	Restricts the cursor to a given rectangle.
<u>CopyCursor</u>	Copies a cursor.
<u>CreateCursor</u>	Creates a cursor from two bit masks.
<u>DestroyCursor</u>	Destroys a cursor created by the <u>CreateCursor</u> function.
<u>GetClipCursor</u>	Retrieves the screen coordinates of the rectangle to which the cursor has been restricted.
<u>GetCursor</u>	Retrieves the handle of the current cursor.
<u>GetCursorPos</u>	Stores the cursor position (in screen coordinates).

<u>LoadCursor</u>	Loads a cursor from the resource file.
<u>SetCursor</u>	Sets the cursor shape.
<u>SetCursorPos</u>	Sets the position of the cursor.
<u>ShowCursor</u>	Increases or decreases the cursor display count.

## Hooks

A hook is a point in the Windows message-handling mechanism that an application can use to gain access to the message stream. Windows provides many types of hooks; each type provides access to a particular type or range of messages. To take advantage of a particular hook, an application can install a filter function that processes the messages associated with the hook. A filter function processes the messages before they reach the destination window procedure.

### Filter-Function Chain

A filter-function chain is a series of connected filter functions for a particular system hook. For example, all keyboard filter functions are installed by `WH_KEYBOARD` and all journaling-record filter functions are installed by `WH_JOURNALRECORD`. An application passes a filter function to a system hook with a call to the `SetWindowsHook` function. Each call adds a new filter function to the beginning of the chain. Whenever an application passes the address of a filter function to a system hook, it must reserve space for the address of the next filter function in the chain. `SetWindowsHook` installs a hook function into a hook chain and returns a handle of the hook.

Once each filter function completes its task, it must call the `DefHookProc` function. `DefHookProc` uses the address stored in the location reserved by the application to access the next filter function in the chain.

To remove a filter function from a filter chain, an application must call the `UnhookWindowsHook` function with the type of hook and a pointer to the function.

The standard window hooks and debugging hooks are as follows:

Type	Purpose
<code>WH_CALLWNDPROC</code>	Installs a window filter.
<code>WH_CBT</code>	Installs a computer-based training (CBT) filter.
<code>WH_DEBUG</code>	Installs a debugging filter.
<code>WH_GETMESSAGE</code>	Installs a message filter (on debugging versions only).
<code>WH_HARDWARE</code>	Installs a nonstandard hardware-message filter.
<code>WH_JOURNALPLAYBACK</code>	Installs a journaling playback filter.
<code>WH_JOURNALRECORD</code>	Installs a journaling record filter.
<code>WH_KEYBOARD</code>	Installs a keyboard filter.
<code>WH_MOUSE</code>	Installs a mouse-message filter.
<code>WH_MSGFILTER</code>	Installs a message filter.
<code>WH_SYSMSGFILTER</code>	Installs a systemwide message filter.

Note: The `WH_CALLWNDPROC` and `WH_GETMESSAGE` hooks will affect system performance. They are supplied for debugging purposes only.

### Installing a Filter Function

To install a filter function, an application must do the following:

- 1 Export the function in its module-definition (.DEF) file.
- 2 Obtain the function's address by using the `GetProcAddress` function. (The `MakeProcInstance` function is used only when the filter function is not in a DLL.)
- 3 Call the `SetWindowsHook` function, specifying the type of hook function and the address of the function (returned by `GetProcAddress`).
- 4 Store the return value from `SetWindowsHook` in a reserved location. This value is the handle of the previous filter function.

Note: Filter functions for system-wide hooks must reside in a dynamic-link library (DLL). Application-specific filter functions can reside in a DLL or the application.

## Property Lists

A property list is a storage area that contains handles for data that the application needs to associate with a window.

### Using Property Lists

Once a data handle is in a window's property list, any application that can access the window can also access the handle. Using the property list is a convenient way to make data (for example, an alternate title or menu for a window) available when the application needs to modify a window.

Every window has its own property list. When a window is created, the list is empty. The SetProp function adds entries to the list. Each entry contains a unique Windows character string and a data handle.

The data handle can identify any object that the application needs to associate with the window. The GetProp function retrieves the data handle of an entry from the list without removing the entry. The handle can then be used to retrieve or use the data. The RemoveProp function removes an entry from the list when it is no longer needed.

Although the purpose of the property list is to associate data with a window for use by the application that owns the window, the handles in a property list are accessible to any application that has access to the window. This means an application can retrieve and use a data handle from the property list of a window created by another application. But using another application's data handles must be done with care. Only shared, global memory objects, such as GDI drawing objects, can be used by other applications. If a property list contains local or global memory handles or resource handles, only the application that has created the window can use them. An application can use the Windows clipboard to share global memory handles with other applications. Local memory handles cannot be shared.

The contents of a property list can be enumerated by using the EnumProps function. The function passes the string and data handle of each entry in the list to an application-supplied function. The application-supplied function can then carry out the necessary task.

The data handles in a property list always belong to the application that created them. The property list itself, like other window-related data, belongs to Windows. A window's property list is allocated in the USER heap, the local heap of the USER library. Although there is no defined limit to the number of entries in a property list, the number of entries depends on how much space is available in the USER heap. The available space depends on how many windows, window classes, and other window-related objects have been created.

The application creates the entries in a property list. Before a window is destroyed or the application that owns the window closes, all entries in the property list must be removed by using the RemoveProp function. Failure to remove the entries leaves the property list in the USER heap and makes the space it occupies unusable for subsequent applications. This can ultimately cause an overflow of the USER heap.

An application can use the RemoveProp function at any time to remove entries from the property list. If there are entries in the property list when the WM\_DESTROY message is received for the window, the entries must be removed at that time. To ensure that all entries are removed, use the EnumProps function to enumerate all entries in the property list. An application should remove only those properties that it added to the property list. Windows adds properties for its own use and disposes of them automatically. An application must not remove properties that Windows has added to the list.

## Rectangles

In Windows, a rectangle is defined by a RECT structure. The structure specifies two points: the upper-left and lower-right corners of the rectangle. The sides of a rectangle extend from these two points and are parallel to the x- and y-axes.

### Using Rectangles in a Windows Application

Rectangles are used to specify rectangular areas on the screen or in a window, such as the cursor clipping region, the client repaint area, a formatting area for formatted text, and the scroll area. Rectangles are also used to fill, frame, or invert an area in the client area with a given brush, and to retrieve the coordinates of a window or a window's client area.

Because rectangles are used for many different purposes, the rectangle functions do not use an explicit unit of measure. Instead, all rectangle coordinates and dimensions are given in signed, logical values. The units of measure are determined by the function in which the rectangle is used.

### Rectangle Coordinates

Valid coordinate values for a rectangle are in the range -32,768 through 32,767. Valid widths and heights, which must be positive, are in the range 0 through 32,767. This means that a rectangle whose left and right sides or whose top and bottom are further apart than 32,768 units is not valid.

### Creating and Manipulating Rectangles

The SetRect function creates a rectangle, the CopyRect function makes a copy of a given rectangle, and the SetRectEmpty function creates an empty rectangle. An empty rectangle is any rectangle that has zero width, zero height, or both.

The InflateRect function increases or decreases the width or height of a rectangle, or both. It can add or remove width from both ends of the rectangle; it can add or remove height from both the top and bottom of the rectangle.

The OffsetRect function moves the rectangle by a given amount. It moves the rectangle by adding the given x-amount, y-amount, or x- and y-amounts to the corner coordinates.

The PtInRect function finds out whether a given point lies within a given rectangle. The point is in the rectangle if it lies on the left or top side or is completely within the rectangle.

The IsRectEmpty function finds out whether the given rectangle is empty.

The IntersectRect function creates a new rectangle that is the intersection of two existing rectangles. The intersection is the largest rectangle contained in both existing rectangles.

The UnionRect function creates a new rectangle that is the union of two existing rectangles. The union is the smallest rectangle that contains both existing rectangles.

For information about functions that draw ellipses and polygons, see Graphics Device Interface Overview.



## Graphics Device Interface Overview

The following topics describe the functions that perform device-independent graphics operations in an application for the Microsoft Windows operating system. These operations include the creation of line, text, and bitmap output on different output devices. The functions performing those operations constitute the Windows graphics device interface (GDI).

Device Contexts

Specifying Colors

Color-palette functions

Drawing-attribute functions

Mapping functions

Coordinate functions

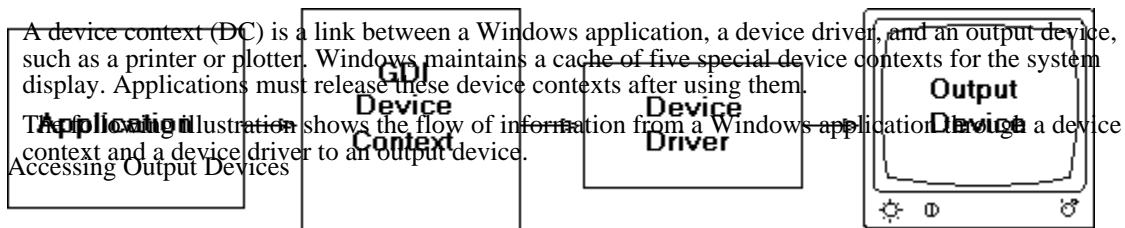
Line-output functions

Ellipse and polygon functions

Metafile functions

Some Windows functions in the USER application programming interface (API) are closely related to these GDI function groups.

## Device Contexts



Any Windows application can use GDI functions to access an output device. GDI passes calls, which are device independent, from the application to the device driver. The device driver then translates the calls into device-dependent operations.

### Saving and Restoring a Device Context

The SaveDC and RestoreDC functions save and restore device contexts. The former saves the original attributes, and the latter makes them available at a later time. For example, a Windows application may need to save its original clipping region so that it can restore the original state of the client area after a series of alterations occur.

### Deleting a Device Context

The DeleteDC function deletes a device context and ensures that shared resources are not removed until the last context is deleted. The device driver is a shared resource. DeleteDC should be used to delete device contexts created by the application. If the application uses the GetDC function to retrieve a device context, it should use the ReleaseDC function, not DeleteDC.

### Creating a Compatible Device Context

The CreateCompatibleDC function causes Windows to treat a portion of memory as a virtual device. Then Windows prepares a device context that has the same attributes as the device for which the virtual device was created, but the device context has no connected output device.

To use the compatible device context, the application creates a compatible bitmap and selects it into the device context. Any output the application sends to the device is drawn in the selected bitmap. Because the device context is compatible with an actual device, the context of the bitmap can be copied directly to the actual device, or vice versa. This also means that the application can send output to memory (prior to sending it to the device).

Note: The CreateCompatibleDC function works only for devices that support raster operations. To discover whether a device supports raster operations, an application can call the GetDeviceCaps function with the RC\_BITBLT index.

### Creating an Information Context

The CreateIC function creates an information context for a device. An information context is a device context with limited capabilities; it cannot be used to write to the device. An application uses an information context to gather information about the selected device. Information contexts are useful in large applications that require memory conservation.

By using an information context and the GetDeviceCaps function, you can obtain the following device information:

- Device technology
- Physical display size
- Color capabilities of the device
- Color-palette capabilities of the device
- Drawing objects available on the device
- Clipping capabilities of the device
- Raster capabilities of the device
- Curve-drawing capabilities of the device
- Line-drawing capabilities of the device
- Polygon-drawing capabilities of the device
- Text capabilities of the device

## Device-Context Attributes

Device-context attributes describe selected drawing objects (pens and brushes), the selected font and its color, the way in which objects are drawn (or mapped) to the device, the area on the device available for output (clipping region), and other important information. The structure that contains the device-context attributes is called the device-context data block. The default attributes and the GDI functions that affect or use them are as follows.

Attribute	Default	GDI functions
Background color	White	<u>SetBkColor</u>
Background mode	OPAQUE	<u>SetBkMode</u>
Bitmap	No default	<u>CreateBitmap</u> <u>CreateBitmapIndirect</u> <u>CreateCompatibleBitmap</u> <u>SelectObject</u>
Brush	WHITE_BRUSH	<u>CreateBrushIndirect</u> <u>CreateDIBPatternBrush</u> <u>CreateHatchBrush</u> <u>CreatePatternBrush</u> <u>CreateSolidBrush</u> <u>SelectObject</u>
Brush origin	(0,0)	<u>SetBrushOrg</u> <u>UnrealizeObject</u>
Clipping region	Display surface	<u>CreateEllipticRgn</u> <u>CreateEllipticRgnIndirect</u> <u>CreatePolygonRgn</u> <u>CreatePolyPolygonRgn</u> <u>CreateRectRgn</u> <u>CreateRoundRectRgn</u> <u>ExcludeClipRect</u> <u>IntersectClipRect</u> <u>OffsetClipRgn</u> <u>SelectClipRgn</u>
Color palette	DEFAULT_PALETTE	<u>CreatePalette</u> <u>RealizePalette</u> <u>SelectPalette</u> <u>UnrealizeObject</u>
Current pen position	(0,0)	<u>LineTo</u> <u>MoveTo</u>
Drawing mode	R2_COPYPEN	<u>SetROP2</u>
Font	SYSTEM_FONT	<u>CreateFont</u> <u>CreateFontIndirect</u> <u>SelectObject</u>
Intercharacter spacing	0	<u>SetTextCharacterExtra</u>
Mapping mode	MM_TEXT	<u>SetMapMode</u>
Pen	BLACK_PEN	<u>CreatePen</u> <u>CreatePenIndirect</u> <u>SelectObject</u>
Polygon-filling mode	ALTERNATE	<u>SetPolyFillMode</u>
Stretching mode	BLACKONWHITE	<u>SetStretchBltMode</u>
Text color	Black	<u>SetTextColor</u>
Viewport extent	(1,1)	<u>SetViewportExt</u>
Viewport origin	(0,0)	<u>SetViewportOrg</u>
Window extent	(1,1)	<u>SetWindowExt</u>
Window origin	(0,0)	<u>SetWindowOrg</u>

## Specifying Colors

Many of the GDI functions that create pens and brushes require that the calling application specify a color in the form of a doubleword. The color can be specified as:

- An explicit RGB value
- An index to a logical-palette entry
- A palette-relative RGB value

The second and third methods of specifying color require the application to create a logical palette.

An explicit RGB doubleword value is a long integer that contains a red, a green, and a blue color field. The first (low-order) byte contains the red field, the second byte contains the green field, the third byte contains the blue field, and the fourth (high-order) byte must be zero. Each field specifies the intensity of the color; zero indicates the lowest intensity, and 255 indicates the highest. For example, 0x00FF0000 specifies pure blue, and 0x0000FF00 specifies pure green. The RGB macro accepts values for the relative intensities of the three colors and returns an explicit RGB doubleword value.

When GDI receives the RGB value as a function parameter, it passes the RGB color value directly to the output device driver, which selects the closest available color on the device. The GetNearestColor function returns the logical color closest to a specified logical color that a given device can represent.

If the device is a plotter, the driver converts the RGB value to a single color that matches one of the pens on the device.

If the device uses color raster technology and the RGB value specifies a color for a pen, the driver selects a solid color. If the device uses color raster technology and the RGB value specifies a color for a brush, the driver selects from a variety of available color combinations. Because many color devices can display only a few colors, the actual color is simulated by dithering (that is, mixing pixels of colors that the device can actually render).

If the device is monochrome (black-and-white), the driver selects black, white, or a shade of gray, depending on the RGB value. If the sum of the RGB values is zero, the driver selects a black brush. If the sum of the RGB values is 765, the driver selects a white brush. If the sum of the RGB values is between zero and 765, the driver selects one of the gray patterns available.

The GetRValue, GetGValue, and GetBValue macros extract the values for red, green, and blue from an explicit RGB doubleword value.

## Color Palettes

Many color graphics displays are capable of displaying a wide range of colors. In most cases, however, the actual number of colors that the display can render at any given time is more limited. For example, a display that is potentially able to produce over 262,000 different colors may be able to show only 256 of those colors at a time because of hardware limitations.

To render colors, a display device often maintains a palette of colors. When an application requests a color that is not currently displayed, the display device adds the requested color to the palette. However, when the number of requested colors exceeds the maximum number for the device, it must replace an existing color with the requested color. As a result, if the total number of colors requested by one or more windows exceeds the number available on the display, many of the actual colors displayed will be incorrect.

Windows color palettes act as a buffer between color-intensive applications and the system. When a window has the input focus, Windows ensures that the window displays all the colors it requests, up to the maximum number simultaneously available on the display, and displays additional colors by matching them to available colors. In addition, Windows matches the colors requested by inactive windows as closely as possible to the available colors. This process significantly reduces undesirable changes in the colors displayed in inactive windows.

### Understanding Color Palettes

Color palettes provide a device-independent method for accessing the color capabilities of a display device by managing the physical, or system, palette of the device, if one is available. Typically, devices that can display at least 256 colors use a system palette.

An application employs the system palette by creating and using one or more logical palettes. Each entry in the system palette contains a specific color. Then, instead of specifying an explicit value for a color when performing graphics operations, the application indicates which color is to be displayed by supplying an index into the logical palette.

Because more than one application can use logical palettes, it is possible that the total number of colors requested for display can exceed the capacity of the display device. Windows acts as a mediator among the applications.

When a window requests that its logical palette be given its requested colors (a process known as realizing its palette), Windows first matches entries in the logical palette to current entries in the system palette. If an exact match for a given logical palette entry is not possible, Windows sets the entry in the logical palette into an unused entry in the system palette.

When all entries in the system palette have been used, Windows takes the logical palette entries that do not exactly match and matches them as closely as possible to entries already in the system palette. To further aid color matching, Windows sets aside 20 static colors in the system palette (the default palette) to which it can match entries in a background palette.

Windows always satisfies the color requests of the foreground window first; this procedure ensures that the active window has the best color display possible. For the remaining windows, Windows satisfies the color requests of the window that most recently received the input focus, the window that was active before that one, and so on.

The following illustration shows this process. In this illustration, a hypothetical display has a system palette capable of containing 12 colors. The application that created Logical Palette 1 owns the active window and was the first to realize its logical palette, which consists of 8 colors. Because the active window was active when it realized its palette, Windows mapped all of the colors in Logical Palette 1 directly to the system palette.

Logical Palette 2 is owned by a window that realized its logical palette while it was inactive. Three of the colors (1, 3, and 5) in Logical Palette 2 were identical to colors in the system palette. To save space in the palette, Windows simply matched those colors to existing system colors when the second application realized its palette. Colors 0, 2, 4, and 6 were not already in the system palette, however, so Windows mapped those colors into the system palette. Because the system palette became full, Windows was not able to map the remaining two colors (which did not exactly match existing colors in the system palette) into the system palette. Instead, it matched them to the closest colors in the system palette.

## Using a Color Palette

Before drawing to the display device with a color palette, an application must first create a logical palette by calling the `CreatePalette` function and then use the `SelectPalette` function to select the palette for the device context of the output device for which it will be used. An application cannot select a palette into a device context by using the `SelectObject` function.

All functions with a color parameter accept an index to an entry in the logical palette. The palette index specifier is a long integer value with the first bit in its high-order byte set to 1 and the palette index in the two low-order bytes. For example, 0x01000005 specifies the palette entry with an index of 5. The `PALETTEINDEX` macro accepts an integer value representing the index of a logical palette entry and returns a palette index value, which an application can use as a parameter for GDI functions that require a color.

An application can also specify a palette index indirectly by using a palette-relative `RGB` value. If the target display device supports logical palettes, Windows matches the palette-relative `RGB` value to the closest palette entry. If the target device does not support palettes, the `RGB` value is used as though it were an explicit `RGB` value. The palette-relative `RGB` value is identical to an explicit `RGB` value except that the second bit of the high-order byte is set to 1. For example, 0x02FF0000 specifies a palette-relative `RGB` value for pure blue. The `PALETTERGB` macro accepts values for red, green, and blue and returns a palette-relative `RGB` value, which an application can use as a parameter for GDI functions that require a color.

If an application specifies an `RGB` value instead of a palette entry, Windows uses the closest matching color in the default palette of 20 static colors.

If the source and destination device contexts have selected and realized different palettes, the `BitBlt` function does not properly move bitmap bits to or from a memory device context. In this case, you must call the `GetDIBits` function with the `DIB_RGB_COLORS` flag to retrieve the bitmap bits from the source bitmap in a device-independent format. Then you use the `SetDIBits` function to set the retrieved bits in the destination bitmap. This ensures that Windows properly matches colors between the two device contexts.

Note: The `BitBlt` function successfully moves bitmap bits between two screen display contexts, even if they have selected and realized different palettes. The `StretchBlt` function properly moves bitmap bits between device contexts whether or not they use different palettes.

## Color-Palette Functions

Windows color palettes allow an application to use as many colors as needed without interfering with its own color display or colors displayed by other windows. Following are the functions an application calls to use color palettes:

Function	Description
<code>AnimatePalette</code>	Replaces entries in a logical palette; Windows maps the new entries into the system palette immediately.
<code>CreatePalette</code>	Creates a logical palette.
<code>GetNearestColor</code>	Retrieves the solid color closest to a specified logical color that a given device can represent.
<code>GetNearestPaletteIndex</code>	Retrieves the index of a logical palette entry most nearly matching a specified <code>RGB</code> value.
<code>GetPaletteEntries</code>	Retrieves entries from a logical palette.
<code>GetSystemPaletteEntries</code>	Retrieves a range of palette entries from the system palette.
<code>GetSystemPaletteUse</code>	Determines whether an application has access to the full system palette.
<code>ResizePalette</code>	Changes the size of the specified logical palette.
<code>SetPaletteEntries</code>	Sets new palette entries in a logical palette; Windows does not map the new entries to the system palette until the application realizes the logical palette.
<code>SetSystemPaletteUse</code>	Allows an application to use the full system palette.
<code>UpdateColors</code>	Performs a pixel-by-pixel translation of each pixel's current color to the system palette. This process allows an inactive window to correct its colors without redrawing its client area.

The USER API also provides two palette-management functions:

Function	Description
<u>RealizePalette</u>	Maps entries in a logical palette to the system palette.
<u>SelectPalette</u>	Selects a logical palette into a device context.

For more information about these USER functions, Window Management.

## Drawing Attributes

A drawing attribute can take one of the following forms: line, brush, text, or bitmap output.

### Setting Colors

Line output can be solid or broken (dashed, dotted, or a combination of the two). If it is broken, the space between the breaks can be filled by setting the background mode to OPAQUE and selecting a color. By setting the background mode to TRANSPARENT, the space between breaks is left in its original state. The SetBkMode and SetBkColor functions set the background mode and color.

Brush output is solid, patterned, or hatched. The space between hatch marks can be filled by setting the background mode to OPAQUE and selecting a color. When Windows creates brush output on a display, it combines the existing color on the display surface with the brush color to yield a new and final color; this is a binary raster operation. If the default raster operation is not appropriate, a new one is chosen by using the SetROP2 function.

The appearance of text output is limited only by the number of available fonts and the color capabilities of the output device. The SetBkColor function sets the color of the text background (the unused portion of each character cell), and the SetTextColor function sets the color of the character itself.

### Controlling Stretch

The appearance of bitmap output can be affected by the stretch mode, which determines how lines eliminated from the bitmap are combined. If an application copies a bitmap to a device and it is necessary to shrink or expand the bitmap before drawing, the effects of the StretchBlt and StretchDIBits functions can be controlled by calling the SetStretchBltMode function to set the current stretch mode for a device context.



## Mapping Modes

To maintain device independence, GDI creates output in a logical space and maps it to the display. The mapping mode defines the relationship between units in the logical space and pixels on a device.

There are eight different GDI mapping modes, each of which has a specific use in a Windows application. Following are these mapping modes:

Mapping mode	Description
MM_ANISOTROPIC	Maps one logical unit to an arbitrary physical unit. The x-axis and y-axis are arbitrarily scaled.
MM_HIENGLISH	Maps one logical unit to 0.001 inch. The positive y-axis extends upward.
MM_HIMETRIC	Maps one logical unit to 0.01 millimeter. The positive y-axis extends upward.
MM_ISOTROPIC	Maps one logical unit to an arbitrary physical unit. One unit along the x-axis is always equal to one unit along the y-axis.
MM_LOENGLISH	Maps one logical unit to 0.01 inch. The positive y-axis extends upward.
MM_LOMETRIC	Maps one logical unit to 0.1 millimeter. The positive y-axis extends upward.
MM_TEXT	Maps one logical unit to one pixel. The positive y-axis extends downward.
MM_TWIPS	Maps one logical unit to 1/1440 inch (1/20 of a point; a point is 1/72 inch). The positive y-axis extends upward.

### Constrained Mapping Modes

GDI classifies six of the mapping modes as constrained mapping modes. These mapping modes are constrained because the scaling factor is fixed, so an application cannot change the number of logical units that Windows maps to a physical unit. The relationship of logical units to physical units for each constrained mapping mode follows:

Mapping mode	Logical units	Physical unit
MM_HIENGLISH	1000	1 inch
MM_HIMETRIC	100	1 millimeter
MM_LOENGLISH	100	1 inch
MM_LOMETRIC	10	1 millimeter
MM_TEXT	1	Device pixel
MM_TWIPS	1440	1 inch

Note: The MM\_HIENGLISH, MM\_HIMETRIC, MM\_LOENGLISH, MM\_LOMETRIC, and MM\_TWIPS mapping modes sometimes map logical units to device units in ways that do not correspond exactly to the preceding table. This typically occurs on displays; for example, on an VGA display there is a 33 percent increase in the dimensions of the device units. The increase in the dimensions of device units occurs so that the same output looks equally crisp and readable whatever the device resolution and the display technology for the device. An application can use the GetDeviceCaps function with the LOGPIXELSX and LOGPIXELSY indices to discover the scaling factor.

### Logical coordinate system

### Physical coordinate system

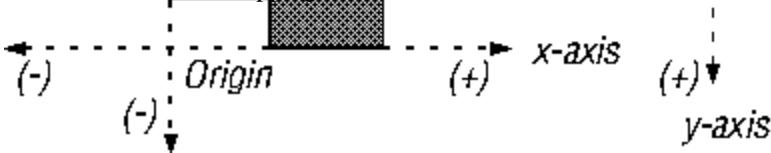
In each of the constrained modes, one logical unit is mapped to a predefined physical unit. For instance, the MM\_TEXT mapping mode maps one logical unit to one device pixel, and the MM\_LOENGLISH mapping mode maps one logical unit to 0.01 inch of the device. Examples for these two modes follow.

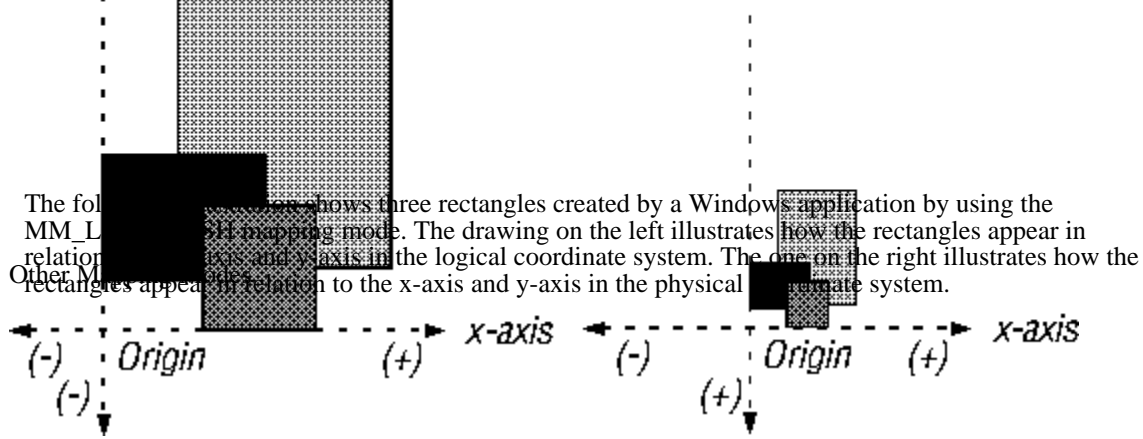
#### MM\_TEXT Mapping Mode

The default mapping mode is MM\_TEXT. In this mapping mode one logical unit is mapped to one pixel on the device or display.

The following illustration shows three rectangles created by a Windows application by using the MM\_TEXT mapping mode. The drawing on the left illustrates the logical coordinate space, and the one on the right illustrates the physical, or physical, coordinate space. The rectangles appear vertically elongated in the logical space because pixels on the chosen display are longer than they are wide. The rectangles appear horizontally elongated in the physical space because the positive y-axis extends downward in the physical-

#### MM\_LOENGLISH Mapping Mode





The MM\_ISOTROPIC and MM\_ANISOTROPIC mapping modes, which are not constrained, use two rectangular regions to derive a scaling factor and an orientation: the window and the viewport. The window lies within the logical-coordinate space, and the viewport lies within the physical-coordinate space. Both possess an origin, an x-extent, and a y-extent. The origin may be any one of the four corners. The x-extent is the horizontal distance from the origin to its opposing corner. The y-extent is the vertical distance from the origin to its opposing corner.

Windows creates a horizontal scaling factor by dividing the viewport's x-extent by the window's x-extent and creates a vertical scaling factor by dividing the viewport's y-extent by the window's y-extent. These scaling factors determine the number of logical units that Windows maps to a number of pixels. In addition to determining scaling factors, the window and viewport determine the orientation of an object. Windows always maps the window origin to the viewport origin, the window x-extent to the viewport x-extent, and the window y-extent to the viewport y-extent.

#### Partially Constrained Mapping Mode

An application creates output with equally scaled axes by using the MM\_ISOTROPIC mapping mode. As the term isotropic implies, Windows maps a symmetrical object (for example, a square or a circle) in the logical space as a symmetrical object in the physical space. In order to maintain this symmetry, GDI shrinks one of the viewport extents. The amount of shrinkage depends on the requested extents and the aspect ratio of the device. This mapping mode is called partially constrained because the application does not have complete control in altering the scaling factor.

#### Unconstrained Mapping Mode

An application can completely alter the horizontal and vertical scaling factors by using the MM\_ANISOTROPIC mapping mode and setting the window and viewport extents to any value after selecting this mapping mode. Windows does not alter either scaling factor in this mode.

## Coordinate Functions

Coordinate functions convert client coordinates to screen coordinates (or vice versa). These functions are useful in graphics-intensive applications. Following are the coordinate functions:

Function	Description
DPtoLP	Converts device points (that is, points relative to the window origin) into logical points.
<u>GetCurrentPosition</u>	Retrieves the current position, in logical coordinates.
<u>GetCurrentPositionEx</u>	Retrieves position in logical units.
LPtoDP	Converts logical points into device points.

GDI uses the following equations to transform logical points to device points and device points to logical points:

- Transforming logical points to device points:

$$Dx = (Lx - xWO) * xVE/xWE + xVO$$

$$Dy = (Ly - yWO) * yVE/yWE + yVO$$

- Transforming device points to logical points:

$$Lx = (Dx - xVO) * xWE/xVE + xWO$$

$$Ly = (Dy - yVO) * yWE/yVE + yWO$$

Following are descriptions of the variables used in these transformation equations:

Variable	Description
xWO	Window origin x-coordinate
yWO	Window origin y-coordinate
xWE	Window extent x-coordinate
yWE	Window extent y-coordinate
xVO	Viewport origin x-coordinate
yVO	Viewport origin y-coordinate
xVE	Viewport extent x-coordinate
yVE	Viewport extent y-coordinate
Lx	Logical-coordinate system x-coordinate
Ly	Logical-coordinate system y-coordinate
Dx	Device x-coordinate
Dy	Device y-coordinate

The following four ratios are scaling factors used to determine the necessary stretching or compressing of logical units:  $xVE/xWE$ ,  $yVE/yWE$ ,  $xWE/xVE$ , and  $yWE/yVE$ .

The subtraction and addition of viewport and window origins is referred to as the translational component of the equation.

In addition, applications can use the following functions from the USER API to convert coordinates from one system to another:

Function	Description
<u>ChildWindowFromPoint</u>	Determines which, if any, of the child windows belonging to a given parent window contains a specified point.
<u>ClientToScreen</u>	Converts the client coordinates of a given point on the display to screen coordinates.
<u>ScreenToClient</u>	Converts the screen coordinates of a given point on the display to client coordinates.
<u>WindowFromPoint</u>	Retrieves the handle of the window that contains a given point.

## Line Output

Line output functions require coordinates in logical units, which GDI uses to draw a line in logical space. (The use of logical units ensures device independence in Windows.) GDI maps this line from the logical space to pixels on the device. The number of logical units that GDI maps to a pixel depends on the current mapping mode. When GDI draws a line, it excludes the last specified point.

If an application draws lines and does not create a new pen, GDI uses the default pen. This pen is black and is one pixel wide when the mapping mode is MM\_TEXT. An application can create a new pen of a different width, style, and color by using the CreatePen function. The new color is dependent on the color capabilities of the output device. The new style can be solid, dotted, dashed, or combined (dotted and dashed). Once an application creates a new pen, it can select the pen into a display context by using the SelectObject function.

## Ellipses and Polygons

Ellipse and polygon functions require coordinates in logical units, which GDI uses to determine the location and size of an object in logical space. (The use of logical units ensures device independence in Windows.) GDI maps the object from logical space to pixels on the device. The number of logical units that Windows maps to a pixel depends on the current mapping mode. The default mapping mode, `MM_TEXT`, maps one logical unit to one pixel.

### Rectangles

The Rectangle function draws a rectangle, using the current pen. The RoundRect function also draws a rectangle, but with rounded rather than square corners.

When GDI draws a rectangle, it uses four arguments. The first two arguments specify the upper-left corner of the rectangle. The last two arguments do not actually specify part of the rectangle; they specify the point adjacent to the lower-right corner. For example, if the first point is specified by  $(x1, y1)$  and the second point is specified by  $(x2, y2)$ , the rectangle's upper-left corner will be  $(x1, y1)$  and the lower-right corner will be  $(x2 - 1, y2 - 1)$ .

### Bounding Rectangles

The Chord, Ellipse, and Pie functions use a bounding rectangle, instead of a radius or circumference measurement, to define the size of the object they create. The bounding rectangle is hidden; GDI uses it only to describe the location and size of the object.

### Ellipse and Polygon Functions

Ellipse and polygon functions, which draw ellipses and polygons, are particularly useful in drawing and charting applications. GDI draws the perimeter of each object with the selected pen and fills the interior by using the selected brush. Following are the ellipse and polygon functions:

Function	Description
<u>Chord</u>	Draws a chord.
<u>Ellipse</u>	Draws an ellipse.
<u>Pie</u>	Draws a pie.
<u>Polygon</u>	Draws a polygon.
<u>PolyPolygon</u>	Draws a series of closed polygons that are filled as though they were a single polygon.
<u>Rectangle</u>	Draws a rectangle.
<u>RoundRect</u>	Draws a rounded rectangle.

## Metafiles

A metafile is a collection of GDI commands that creates desired text or images. Metafiles provide a convenient method of storing graphics commands that create text or images. Metafiles are especially useful in applications that use specific text or a particular image repeatedly. They are also device-independent; by creating text or images with GDI commands and then placing the commands in a metafile, an application can re-create the text or images repeatedly on a variety of devices. Metafiles are also useful in applications that need to pass graphics information to other applications.

### Creating a Metafile

A Windows application must create a metafile in a special device context. It cannot use the device contexts that the CreateDC or GetDC function returns; instead, it must use the device context that the CreateMetaFile function returns.

Windows allows an application to use a subset of the GDI functions to create a metafile. This subset consists of all GDI functions that create output (rather than functions that provide state information, such as the GetDeviceCaps function). The following list shows GDI functions that an application can use in a metafile:

<u>AnimatePalette</u>	<u>OffsetViewportOrg</u>	<u>SetBkMode</u>
<u>Arc</u>	<u>OffsetWindowOrg</u>	<u>SetDIBitsToDevice</u>
<u>BitBlt</u>	<u>PatBlt</u>	<u>SetMapMode</u>
<u>Chord</u>	<u>Pie</u>	<u>SetMapperFlags</u>
<u>CreateBrushIndirect</u>	<u>Polygon</u>	<u>SetPixel</u>
<u>CreateDIBPatternBrush</u>	<u>Polyline</u>	<u>SetPolyFillMode</u>
<u>CreateFontIndirect</u>	<u>PolyPolygon</u>	<u>SetROP2</u>
<u>CreatePatternBrush</u>	<u>RealizePalette</u>	<u>SetStretchBltMode</u>
<u>Ellipse</u>	<u>RestoreDC</u>	<u>SetTextColor</u>
<u>Escape</u>	<u>RoundRect</u>	<u>SetTextJustification</u>
<u>ExcludeClipRect</u>	<u>SaveDC</u>	<u>SetViewportExt</u>
<u>ExtTextOut</u>	<u>ScaleViewportExt</u>	<u>SetViewportOrg</u>
<u>FloodFill</u>	<u>ScaleWindowExt</u>	<u>SetWindowExt</u>
<u>IntersectClipRect</u>	<u>SelectClipRgn</u>	<u>SetWindowOrg</u>
<u>LineTo</u>	<u>SelectObject</u>	<u>StretchBlt</u>
<u>MoveTo</u>	<u>SelectPalette</u>	<u>StretchDIBits</u>
<u>OffsetClipRgn</u>	<u>SetBkColor</u>	<u>TextOut</u>

To create output in a metafile, an application must follow four steps:

- 1 Create a special device context by using the CreateMetaFile function.
- 2 Send GDI commands to the metafile by using the special device context.
- 3 Close the metafile by calling the CloseMetaFile function. This function returns a metafile handle.
- 4 Display the image or text on a device by using the PlayMetaFile function and passing to the function the metafile handle obtained from CloseMetaFile and a device-context handle for the device on which the metafile is to be played.

The device context that the CreateMetaFile function creates does not have default attributes of its own. Whatever device-context attributes are in effect for the output device when an application plays a metafile will be the defaults for the metafile. The metafile can change these attributes while it is playing. If the application needs to retain the same device-context attributes after the metafile has finished playing, it should save the output device context by calling the SaveDC function before calling the PlayMetaFile function. Then, when PlayMetaFile returns, the application can call the RestoreDC function to restore the original device-context attributes.

Although the maximum size of a metafile is  $2^{32}$  bytes or records, the actual size of a metafile is limited by the amount of memory or disk space available.

### Storing a Metafile

An application can store a metafile in system memory or in a disk file.

To store the metafile in memory, an application calls the CreateMetaFile function and passes NULL as the function parameter. The application can free the memory that Windows uses to store the metafile by calling the DeleteMetaFile function. This function removes a metafile from memory and invalidates its handle. DeleteMetaFile has no effect on disk files.

There are two ways of storing a metafile in a disk file:

- When the application calls the CreateMetaFile function to open a metafile, it passes a filename as the function parameter; the metafile is then recorded in a disk file.
- After the application has created a metafile in memory, it calls the CopyMetaFile function. This function accepts the handle of a memory metafile and the name of the disk file to which the metafile will be saved.

The GetMetaFile function opens a metafile stored in a disk file and makes it available for replay or modification. This function accepts the filename of a metafile stored on disk and returns a metafile handle.

#### Changing How Windows Plays a Metafile

A metafile does not have to be played back in its entirety or exactly in the form in which it was recorded. An application can use the EnumMetaFile function to locate a specific metafile record. EnumMetaFile calls a callback function supplied by the application and passes it the following information:

- The metafile device context
- A pointer to the metafile handle table
- A pointer to a metafile record
- The number of associated objects with handles in the handle table
- A pointer to application-supplied data

The callback function can then use this information to play a single record, to query the record, to copy it, or to modify it.

The PlayMetaFileRecord function plays a metafile record by executing the GDI function contained in the record.

When Windows plays or enumerates the records in a metafile, it identifies each object with an index into a handle table. Functions that select objects (such as SelectObject and SelectPalette) identify the object by means of the object handle that the application passes to the function.

Objects are added to the table in the order in which they are created. For example, if a brush is the first object created in a metafile, the brush is given index 0. If the second object is a pen, it is given index 1, and so on. For information about the format of the handle table, see the description of the HANDLETABLE structure in the Microsoft Windows Programmer's Reference, Volume 3.

## Common Dialog Box Overview (3.1)

Common dialog boxes make it easier for you to develop applications for the Microsoft Windows operating system. A common dialog box is a dialog box that an application displays by calling a single function rather than by creating a dialog box procedure and a resource file containing a dialog box template. The dynamic-link library COMMDLG.DLL provides a default procedure and template for each type of common dialog box. Each default dialog box procedure processes messages and notifications for a common dialog box and its controls. A default dialog box template defines the appearance of a common dialog box and its controls.

In addition to simplifying the development of Windows applications, a common dialog box assists users by providing a standard set of controls for performing certain operations. As Windows developers begin using the common dialog boxes in their applications, users will find that after they master using a common dialog box in one application, they can easily perform the same operations in other applications.

This topic describes the various common dialog boxes and includes sample code to help you use common dialog boxes in your Windows applications.

Following are the types of common dialog boxes in the order in which they are presented in this topic:

Name	Description
<u>Color</u>	Displays available colors, from which the user can select one; displays controls that let the user define a custom color.
<u>Font</u>	Displays lists of fonts, point sizes, and colors that correspond to available fonts; after the user selects a font, the dialog box displays sample text rendered with that font.
<u>Open</u>	Displays a list of filenames matching any specified extensions, directories, and drives. By selecting one of the listed filenames, the user indicates which file an application should open.
<u>Save As</u>	Displays a list of filenames matching any specified extensions, directories, and drives. By selecting one of the listed filenames, the user indicates which file an application should save.
<u>Print</u>	Displays information about the installed printer and its configuration. By altering and selecting controls in this dialog box, the user specifies how output should be printed and starts the printing process.
<u>Print Setup</u>	Displays the current list of available printers. The user can select a printer from this list. This common dialog box also provides options for setting the paper orientation, size, and source (when the printer driver supports these options). In addition to being called directly, the Print Setup dialog can be opened from within the Print dialog.
<u>Find</u>	Displays an edit control in which the user can type a string for which the application should search. The user can specify the direction of the search, whether the application should match the case of the specified string, and whether the string to match is an entire word.
<u>Replace</u>	Displays two edit controls in which the user can type strings: the first string identifies a word or value that the application should replace, and the second string identifies the replacement word or value.

Applications that use the common dialog boxes should specify at least 8K for the stack size, as shown in the following example:

```
NAME cd
EXETYPE WINDOWS
STUB 'WINSTUB.EXE'
CODE PRELOAD MOVEABLE DISCARDABLE
DATA PRELOAD MOVEABLE MULTIPLE
HEAPSIZE 1024
STACKSIZE 8192
EXPORTS
    FILEOPENHOOKPROC @1
```



See Also  
Customizing Common Dialog Boxes

## Color Dialog Box (3.1)

### Using Color Dialog Boxes

The Color dialog box contains controls that make it possible for a user to select and create colors.

Following is a Color dialog box.

- The Basic Colors control displays up to 48 colors. The actual number of colors displayed is determined by the display driver. For example, a VGA driver displays 48 colors, and a monochrome display driver displays only 16. With the Basic Colors control, the user can select a displayed color.

To display the Custom Colors control, the user clicks the Define Custom Colors button. The Custom Colors control displays custom colors. The user can select one of the 16 rectangles in this control and then create a new color by using one of the following methods:

- Specifying red, green, and blue (RGB) values by using the Red, Green, and Blue edit controls, and then choosing the Add to Custom Colors button to display the new color in the selected rectangle.
- Moving the cursor in the color spectrum control (at the upper-right of the dialog box) to select hue and saturation values; moving the cursor in the luminosity control (the rectangle to the right of the spectrum control); and then choosing the Add to Custom Colors button to display the new color in the selected rectangle.
- Specifying hue, saturation, and luminosity (HSL) values by using the Hue, Sat, and Lum edit controls and then choosing the Add to Custom Colors button to display the new color in the selected rectangle.

The Color Solid control displays the dithered and solid colors that correspond to the user's selection. (A dithered color is a color created by combining one or more pure or solid colors.) The Flags member of the CHOOSECOLOR structure contains a flag bit that, when set, displays a Help button.

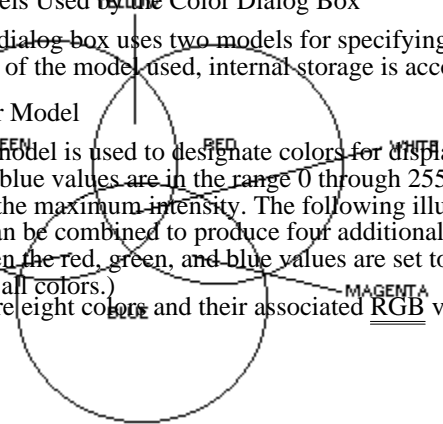
An application can display the Color dialog box in one of two ways: fully open or partially open. When the Color dialog box is displayed partially open, the user cannot change the custom colors.

### Color Models Used by the Color Dialog Box

The Color dialog box uses two models for specifying colors: the RGB model and the HSL model. Regardless of the model used, internal storage is accomplished by use of the RGB model.

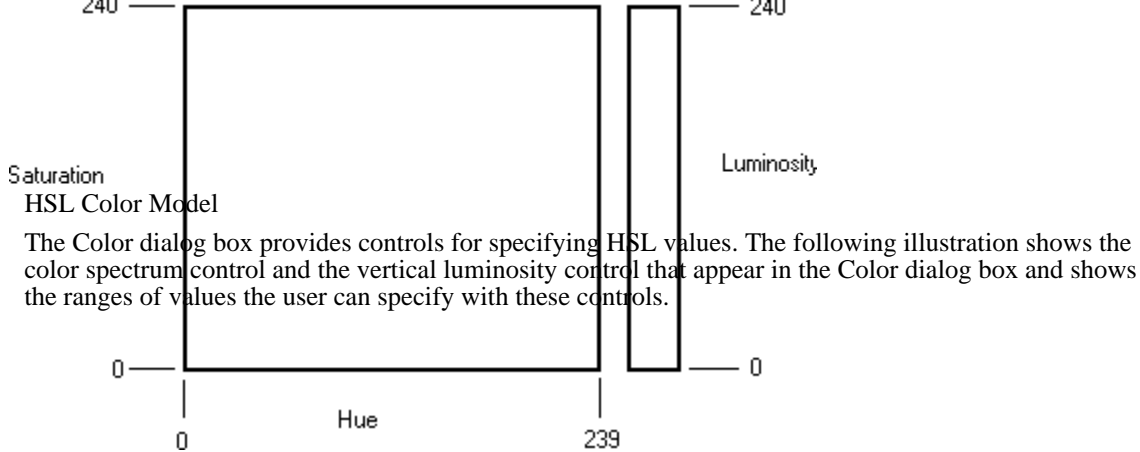
#### RGB Color Model

The RGB model is used to designate colors for displays and other devices that emit light. Valid red, green, and blue values are in the range 0 through 255, with 0 indicating the minimum intensity and 255 indicating the maximum intensity. The following illustration shows how the primary colors red, green, and blue can be combined to produce four additional colors. (With display devices, the color black results when the red, green, and blue values are set to 0--that is, with display technology, black is the absence of all colors.)  
Following are eight colors and their associated RGB values:



Color	RGB values
Red	255, 0, 0
Green	0, 255, 0
Blue	0, 0, 255
Cyan	0, 255, 255
Magenta	255, 0, 255
Yellow	255, 255, 0
White	255, 255, 255
Black	0, 0, 0

Windows stores internal colors as 32-bit RGB values. The high-order byte of the high-order word is reserved; the low-order byte of the high-order word specifies the intensity of the blue component; the high-order byte of the low-order word specifies the intensity of the green component; and the low-order byte of the low-order word specifies the intensity of the red component.



In the Color dialog box, the saturation and luminosity values must be in the range 0 through 240 and the hue value must be in the range 0 through 239.

#### Converting HSL Values to RGB Values

The dialog box procedure provided in `COMMDLG.DLL` for the Color dialog box contains code that converts HSL values to the corresponding RGB values. Following are several colors with their associated HSL and RGB values:

Color	HSL values	RGB values
Red	(0, 240, 120)	(255, 0, 0)
Yellow	(40, 240, 120)	(255, 255, 0)
Green	(80, 240, 120)	(0, 255, 0)
Cyan	(120, 240, 120)	(0, 255, 255)
Blue	(160, 240, 120)	(0, 0, 255)
Magenta	(200, 240, 120)	(255, 0, 255)
White	(0, 0, 240)	(255, 255, 255)
Black	(0, 0, 0)	(0, 0, 0)

#### Using the Color Dialog Box to Display Basic Colors

An application can display the Color dialog box so that a user can select one color from a list of basic screen colors. This section describes how you can provide code and structures in your application that make this possible.

#### Initializing the `CHOOSECOLOR` Structure

Before you display the Color dialog box you need to initialize a `CHOOSECOLOR` structure. This structure should be global or declared as a static variable. The members of this structure contain information about such items as the following:

- Structure size
- Which window owns the dialog box
- Whether the application is customizing the common dialog box
- The hook function and custom dialog box template to use for a customized version of the Color dialog box
- RGB values for the selected basic color

If your application does not customize the dialog box and you want the user to be able to select a single color from the basic colors, you should initialize the `CHOOSECOLOR` structure in the following manner:

```

/* Color variables */

CHOOSECOLOR cc;
COLORREF clr;
COLORREF aclrCust[16];
int i;

/* Set the custom color controls to white. */

for (i = 0; i < 16; i++)
    aclrCust[i] = RGB(255, 255, 255);

/* Initialize clr to black. */

```

```

clr = RGB(0, 0, 0);

/* Set all structure fields to zero. */

memset(&cc, 0, sizeof(CHOOSECOLOR));

/* Initialize the necessary CHOOSECOLOR members. */

cc.lStructSize = sizeof(CHOOSECOLOR);
cc.hwndOwner = hwnd;
cc.rgbResult = clr;
cc.lpCustColors = aclrCust;
cc.Flags = CC_PREVENTFULLOPEN;

if (ChooseColor(&cc))
    . /* Use cc.rgbResult to select the user-requested color. */
    .

```

In the previous example, the array to which the lpCustColors member points contains 16 doubleword RGB values that specify the color white, and the CC\_PREVENTFULLOPEN flag is set in the Flags member to disable the Define Custom Colors button and prevent the user from selecting a custom color.

#### Calling the ChooseColor Function

After you initialize the structure, you should call the ChooseColor function. If the function is successful and the user chooses the OK button to close the dialog box, the rgbResult member contains the RGB values for the basic color that the user selected.

#### Using the Color Dialog Box to Display Custom Colors

An application can display the Color dialog box so that the user can create and select a custom color. This section describes how you can provide code and structures in your application that make this possible.

#### Initializing the CHOOSECOLOR Structure

Before you display the Color dialog box, you need to initialize a CHOOSECOLOR structure. This structure should be global or declared as a static variable. The members of this structure contain information about such items as the following:

- Structure size
- Which window owns the dialog box
- Whether the application is customizing the common dialog box
- The hook function and custom dialog box template to use for a customized version of the Color dialog box
- RGB values for the custom color control

If your application does not customize the dialog box and you want the user to be able to create and select custom colors, you should initialize the CHOOSECOLOR structure in the following manner:

```

/* Color Variables */

CHOOSECOLOR chsclr;
DWORD dwCustClrs[16] = { RGB(255, 255, 255), RGB(239, 239, 239),
RGB(223, 223, 223), RGB(207, 207, 207),
RGB(191, 191, 191), RGB(175, 175, 175),
RGB(159, 159, 159), RGB(143, 143, 143),
RGB(127, 127, 127), RGB(111, 111, 111),
RGB(95, 95, 95), RGB(79, 79, 79),
RGB(63, 63, 63), RGB(47, 47, 47),
RGB(31, 31, 31), RGB(15, 15, 15)
};
BOOL fSetColor = FALSE;
int i;

chsclr.lStructSize = sizeof (CHOOSECOLOR);
chsclr.hwndOwner = hwnd;
chsclr.hInstance = NULL;

```

```

chsclr.rgbResult = 0L;
chsclr.lpCustColors = (LPDWORD) dwCustClrs;
chsclr.Flags = CC_FULLOPEN;
chsclr.lCustData = 0L;
chsclr.lpfHook = (FARPROC) NULL;
chsclr.lpTemplateName = (LPSTR) NULL;

```

In the previous example, the array to which `lpCustColors` points contains sixteen 32-bit RGB values that specify 16 scales of gray, and the `CC_FULLOPEN` flag is set in the `Flags` member to display the complete Color dialog box.

#### Calling the ChooseColor Function

After you initialize the structure, you should call the ChooseColor function as shown in the following code fragment:

```

if (fSetColor = ChooseColor(&chsclr))
.
. /* Use chsclr.lpCustColors to select user specified colors*/
.

```

If the function is successful and the user chooses the OK button to close the dialog box, the `lpCustColors` member points to an array that contains the RGB values for the custom colors requested by the application's user.

Applications can exercise more control over custom colors by creating a new message identifier for the string defined by the `COLOROKSTRING` constant. The application creates the new message identifier by calling the RegisterWindowMessage function and passing this constant as the single parameter. After calling RegisterWindowMessage, the application receives a message immediately prior to the dismissal of the dialog box. The `lParam` parameter of this message contains a pointer to the CHOOSECOLOR structure. The application can use the `lpCustColors` member of this structure to check the current color. If the application returns a nonzero value when it processes this message, the dialog box is not dismissed.

Similarly, applications can create a new message identifier for the string defined by the `SETRGBSTRING` constant. The application's hook function can use the message identifier returned by calling RegisterWindowMessage with the `SETRGBSTRING` constant to set a color in the dialog box. For example, the following line of code sets the color selection to blue:

```

SendMessage(hwndDlg, wSetRGBMsg, 0, (LPARAM) RGB(0, 0, 255));

```

In this example, `wSetRGBMsg` is the message identifier returned by the RegisterWindowMessage function. The `lParam` parameter of the SendMessage function is set to the RGB values of the desired color. The `wParam` parameter is not used.

The application can specify any valid RGB values in this call to SendMessage. If the RGB values match one of the basic colors, the system selects the basic color and updates the spectrum and luminosity controls. If the RGB values do not match one of the basic colors, the system updates the spectrum and luminosity controls, but the basic color selection remains unchanged.

Note that if the Color dialog box is not fully open and the application sends RGB values that do not match one of the basic colors, the system does not update the dialog box. Updates are unnecessary because the spectrum and luminosity controls are not visible when the dialog box is only partially open.

For more information about processing registered window messages, see Using Find and Replace Dialog Boxes.

## Font Dialog Box (3.1)

### Using Font Dialog Boxes

The Font dialog box contains controls that make it possible for a user to select a font, a font style (such as bold, italic, or regular), a point size, and an effect (such as underline, strikeout, or a text color).

Following is a Font dialog box.

#### ■ Displaying the Font Dialog Box in Your Application

The Font dialog box appears after you initialize the members in a CHOOSEFONT structure and call the ChooseFont function. This structure should be global or declared as a static variable. The members of the CHOOSEFONT structure contain information about such items as the following:

- The attributes of the font that initially is to appear in the dialog box.
- The attributes of the font that the user selected.
- The point size of the font that the user selected.
- Whether the list of fonts corresponds to a printer, a screen, or both.
- Whether the available fonts listed are TrueType only.
- Whether the Effects box should appear in the dialog box.
- Whether dialog box messages should be processed by an application-supplied hook function.
- Whether the point sizes of the selectable fonts should be limited to a specified range.
- Whether the dialog box should display only what-you-see-is-what-you-get (WYSIWIG) fonts. (These fonts are resident on both the screen and the printer.)
- The color that the ChooseFont function should use to render text in the Sample box the first time the application displays the dialog box.
- The color that the user selected for text output.

To display the Font dialog box, an application should perform the following steps:

- 1 If the application requires printer fonts, retrieve a device-context handle for the printer and use this handle to set the hDC member of the CHOOSEFONT structure. (If the Font dialog box displays only screen fonts, this member should be set to NULL.)
- 2 Set the appropriate flags in the Flags member of the CHOOSEFONT structure. This setting must include CF\_SCREENFONTS, CF\_PRINTERFONTS, or CF\_BOTH.
- 3 Set the rgbColors member of the CHOOSEFONT structure if the default color (black) is not appropriate.
- 4 Set the nFontType member of the CHOOSEFONT structure using the appropriate constant.
- 5 Set the nSizeMin and nSizeMax members of the CHOOSEFONT structure if the CF\_LIMITSIZE value is specified in the Flags member.
- 6 Call the ChooseFont function.

The following example initializes the CHOOSEFONT structure and calls the ChooseFont function:

```
LOGFONT lf;  
CHOOSEFONT cf;  
  
/* Set all structure fields to zero. */  
  
memset(&cf, 0, sizeof(CHOOSEFONT));  
  
cf.lStructSize = sizeof(CHOOSEFONT);  
cf.hwndOwner = hwnd;  
cf.lpLogFont = &lf;  
cf.Flags = CF_SCREENFONTS | CF_EFFECTS;  
cf.rgbColors = RGB(0, 255, 255); /* light blue */  
cf.nFontType = SCREEN_FONTTYPE;  
  
ChooseFont(&cf);
```

When the user closes the Font dialog box by choosing the OK button, the ChooseFont function returns information about the selected font in the LOGFONT structure to which the lpLogFont member points. An application can use this LOGFONT structure to select the font that will be used to render text. The following example selects a font by using the LOGFONT structure and renders a string of text:

```
hdc = GetDC(hwnd);
```

```
hFont = CreateFontIndirect(cf.lpLogFont);  
hFontOld = SelectObject(hdc, hFont);  
TextOut(hdc, 50, 150,  
"AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz", 52);  
SelectObject(hdc, hFontOld);  
DeleteObject(hFont);  
ReleaseDC(hwnd, hdc);
```

An application can also use the WM\_CHOOSEFONT\_GETLOGFONT message to retrieve the current LOGFONT structure for the Font dialog box before the user closes the dialog box.

## Filename Dialog Boxes (3.1)

### Using Open and Save As Dialog Boxes

The Open dialog box and the Save As dialog box are similar in appearance. Each contains controls that make it possible for the user to specify the location and name of a file or set of files. In the case of the Open dialog box, the user selects the file or files to be opened; in the case of the Save As dialog box, the user selects the file or files to be saved.

### Displaying the Open Dialog Box in Your Application

The Open dialog box appears after you initialize the members of an OPENFILENAME structure and call the GetOpenFileName function.

Following is an Open dialog box.

Before the call to GetOpenFileName, structure members contain such data as the name of the directory and the filter that are to appear in the dialog box. (A filter is a filename extension. The common dialog box code uses the extension to filter appropriate filenames from a directory.) After the call, structure members contain such data as the name of the selected file and the number of characters in that filename.

To display an Open dialog box, an application should perform the following steps:

- 1 Store the valid filters in a character array.
- 2 Set the lpstrFilter member to point to this array.
- 3 Set the nFilterIndex member to the value of the index that identifies the default filter.
- 4 Set the lpstrFile member to point to an array that contains the initial filename and receives the selected filename.
- 5 Set the nMaxFile member to the value that specifies the length of the filename array.
- 6 Set the lpstrFileTitle member to point to a buffer that receives the title of the selected file.
- 7 Set the nMaxFileTitle member to specify the length of the buffer.
- 8 Set the lpstrInitialDir member to point to a string that specifies the initial directory. (If this member does not point to a valid string, it must be set to 0 or point to a string that is set to NULL.)
- 9 Set the lpstrTitle member to point to a string specifying the name that should appear in the title bar of the dialog box. (If this pointer is NULL, the title will be Open.)
- 10 Initialize the lpstrDefExt member to point to the default extension. (This extension can be 0, 1, 2, or 3 characters long.)
- 11 Call the GetOpenFileName function.

The following example initializes an OPENFILENAME structure, calls the GetOpenFileName function, and opens the file by using the lpstrFile member of the structure. The OPENFILENAME structure should be global or declared as a static variable.

```
OPENFILENAME ofn;
char szDirName[256];
char szFile[256], szFileTitle[256];
UINT i, cbString;
char chReplace; /* string separator for szFilter */
char szFilter[256];
HFILE hf;

/* Get the system directory name, and store in szDirName */

GetSystemDirectory(szDirName, sizeof(szDirName));
szFile[0] = '\0';

if ((cbString = LoadString(hinst, IDS_FILTERSTRING,
    szFilter, sizeof(szFilter))) == 0) {
    ErrorHandler();
    return 0L;
}
chReplace = szFilter[cbString - 1]; /* retrieve wildcard */

for (i = 0; szFilter[i] != '\0'; i++) {
```



```

        if (szFilter[i] == chReplace)
            szFilter[i] = '\\0';
    }

    /* Set all structure members to zero. */
    memset(&ofn, 0, sizeof(OPENFILENAME));

    ofn.lStructSize = sizeof(OPENFILENAME);
    ofn.hwndOwner = hwnd;
    ofn.lpstrFilter = szFilter;
    ofn.nFilterIndex = 1;
    ofn.lpstrFile = szFile;
    ofn.nMaxFile = sizeof(szFile);
    ofn.lpstrFileTitle = szFileTitle;
    ofn.nMaxFileTitle = sizeof(szFileTitle);
    ofn.lpstrInitialDir = szDirName;
    ofn.Flags = OFN_SHOWHELP | OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;

    if (GetOpenFileName(&ofn)) {
        hf = lopen(ofn.lpstrFile, OF_READ);
        . /* Perform file operations. */
        .
    }
    else
        ErrorHandler();

```

The string referred to by the `IDS_FILTERSTRING` constant in the preceding example is defined as follows in the resource-definition file:

```

STRINGTABLE
BEGIN
    IDS_FILTERSTRING    "Write Files(*.WRI)|*.wri|Word Files(*.DOC)|*.doc|"
END

```

The vertical bars in this string are used as wildcards. After using the `LoadString` function to retrieve the string, the wildcards are replaced with `NULL`. The wildcard can be any unique character and must be included as the last character in the string. Initializing strings in this manner guarantees that the parts of the string are contiguous in memory and that the string is terminated with two null characters.

Applications that can open files over a network can create a new message identifier for the string defined by the `SHAREVISTRING` constant. The application creates the new message identifier by calling the `RegisterWindowMessage` function and passing this constant as the single parameter. After calling `RegisterWindowMessage`, the application is notified whenever a sharing violation occurs during a call to the `OpenFile` function. For more information about processing registered window messages, see [Using Find and Replace Dialog Boxes](#).

#### Displaying the Save As Dialog Box in Your Application

The Save As dialog box appears after you initialize the members of an `OPENFILENAME` structure and call the `GetSaveFileName` function.

Following is a Save As dialog box.

Before the call to `GetSaveFileName`, structure members contain such data as the name of the initial directory and a filter string. After the call, structure members contain such data as the name of the file to be saved and the number of characters in that filename.

The following example initializes an `OPENFILENAME` structure, calls `GetSaveFileName` function, and saves the file. The `OPENFILENAME` structure should be global or declared as a static variable.

```

OPENFILENAME ofn;
char szDirName[256];
char szFile[256], szFileTitle[256];
UINT i, cbString;
char chReplace; /* string separator for szFilter */

```

```

char szFilter[256];
HFILE hf;

/*
 * Retrieve the system directory name, and store it in
 * szDirName.
 */

GetSystemDirectory(szDirName, sizeof(szDirName));

if ((cbString = LoadString(hinst, IDS_FILTERSTRING,
    szFilter, sizeof(szFilter))) == 0) {
    ErrorHandler();
    return 0;
}

chReplace = szFilter[cbString - 1]; /* retrieve wildcard */

for (i = 0; szFilter[i] != '\0'; i++) {
    if (szFilter[i] == chReplace)
        szFilter[i] = '\0';
}

/* Set all structure members to zero. */

memset(&ofn, 0, sizeof(OPENFILENAME));

/* Initialize the OPENFILENAME members. */

szFile[0] = '\0';

ofn.lStructSize = sizeof(OPENFILENAME);
ofn.hwndOwner = hwnd;
ofn.lpstrFilter = szFilter;
ofn.lpstrFile = szFile;
ofn.nMaxFile = sizeof(szFile);
ofn.lpstrFileTitle = szFileTitle;
ofn.nMaxFileTitle = sizeof(szFileTitle);
ofn.lpstrInitialDir = szDirName;
ofn.Flags = OFN_SHOWHELP | OFN_OVERWRITEPROMPT;

if (GetSaveFileName(&ofn)) {
    .
    . /* Perform file operations. */
    .
}
else
    ErrorHandler();

```

The string referred to by the `IDS_FILTERSTRING` constant in the preceding example is defined in the resource-definition file. It is used in exactly the same way as the `IDS_FILTERSTRING` constant discussed in *Displaying the Open Dialog Box in Your Application*.

#### Monitoring List Box Controls in an Open or Save As Dialog Box

An application can monitor list box selections in order to process and display data in custom controls. For example, an application can use a custom control to display the total length, in bytes, of all of the files selected in the File Name box. One method the application can use to obtain this value is to recompute the total count of bytes each time the user selects a file or cancels the selection of a file. A faster method is for the application to use the `LBSELCHSTRING` message to identify a new selection and add the corresponding file length to the value that appears in the custom control. (Note that in this example, the custom control is a standard Windows control that you identify in a resource file template for one of the common dialog boxes.)

An application registers the selection-change message with the `RegisterWindowMessage` function. Once the application registers the message, it uses this function's return value to identify messages from the

dialog box. The message is processed in the application-supplied hook function for the common dialog box. The wParam parameter of each message identifies the list box in which the selection occurred. The low-order word of the lParam parameter identifies the list box item. The high-order word of the lParam parameter is one of the following values:

Value	Meaning
CD_LBSELCHANGE	Specifies that the item identified by the low-order word of lParam was the item in a single-selection list box.
CD_LBSELSUB	Specifies that the item identified by the low-order word of lParam is no longer selected in a multiple-selection list box.
CD_LBSELADD	Specifies that the item identified by the low-order word of lParam was selected from a multiple-selection list box.
CD_LBSELNOITEMS	Specifies that no items exist in a multiple-selection list box.

For an example that registers a common dialog box message, see [Find and Replace Dialogs](#).

#### Monitoring Filenames in an Open or Save As Dialog Box

Applications can alter the normal processing of an Open or Save As dialog box by monitoring which filename the user types and by performing other, unique operations. For example, one application could prevent the user from closing the dialog box if the selected filename is prohibited; another application could make it possible for the user to select multiple filenames.

To monitor filenames, an application should register the FILEOKSTRING message. An application registers this message by calling the [RegisterWindowMessage](#) function and passing the message name as its single parameter. After the message is registered, the dialog box procedure in COMMCTL32.DLL uses it to signal that the user has selected a filename and chosen the OK button and that the dialog box has checked the filename and is ready to return. The dialog box procedure signals these actions by sending the message to the application's hook function. After receiving the message, the hook function should return a value to the dialog box procedure that called it. If the hook function did not process the message, it should return 0; if the hook function did process the message and the dialog box should close, the hook function should return 0; if the hook function did process the message but the dialog box should not close, the hook function should return 1. (All other return values are reserved.)

## Print Dialog Box (3.1)

### Using Print and Print Setup Dialog Boxes

A Print dialog box contains controls that let a user configure a printer for a particular print job. The user can make such selections as print quality, page range, and number of copies (if the printer supports multiple copies).

Following is a Print dialog box.

■ Choosing the Setup button in the Print dialog box displays the following Print Setup dialog box for a PostScript printer.

The Print Setup dialog box provides controls that make it possible for the user to reconfigure the selected printer.

### Device Drivers and the Print Dialog Box

The Print dialog box differs from other common dialog boxes in that part of its dialog box procedure resides in `COMMDLG.DLL` and part in a printer driver. A printer driver is a program that configures a printer, converts graphics device interface (GDI) commands to low-level printer commands, and stores commands for a particular print job in a printer's queue.

A printer driver exports a function called `ExtDeviceMode`, which displays a dialog box and its controls. In previous versions of Windows, an application called the `LoadLibrary` function to load a device driver and the `GetProcAddress` function to obtain the address of the `ExtDeviceMode` function. This is no longer necessary with the Windows common dialog box interface. Instead of calling `LoadLibrary` and `GetProcAddress`, a Windows application can call a single function, `PrintDlg`, to display the Print dialog box and begin a print job. The code for `PrintDlg` resides in `COMMDLG.DLL`. The dialog box that appears when an application calls `PrintDlg` differs slightly from the dialog box that appears when the application calls directly into the device driver. The functionality is very similar in spite of the different appearance.

### Displaying a Print Dialog Box for the Default Printer

To display a Print dialog box for the default printer, an application must initialize a `PRINTDLG` structure and then call the `PrintDlg` function.

The members of the `PRINTDLG` structure can contain information about such items as the following:

- The printer device context
- Values that should appear in the dialog box controls
- The hook function and custom dialog box template to use for a customized version of the Print dialog box or Print Setup dialog box

An application can display a Print dialog box for the currently installed printer by performing the following steps:

- 1 Setting the `PD_RETURNDC` flag in the `Flags` member of the `PRINTDLG` structure. (This flag should only be set if the application requires a device-context handle.)
- 2 Initializing the `lStructSize`, `hDevMode`, and `hDevNames` members.
- 3 Calling the `PrintDlg` function and passing a pointer to the `PRINTDLG` structure just initialized.

Setting the `PD_RETURNDC` flag causes `PrintDlg` to display the Print dialog box and return a handle identifying a printer device context in the `hDC` member of the `PRINTDLG` structure. (The application passes the device-context handle as the first parameter to the GDI functions that render output on the printer.)

The following example initializes the members of the `PRINTDLG` structure and calls the `PrintDlg` function prior to printing output. This structure should be global or declared as a static variable.

```
PRINTDLG pd;  
  
/* Set all structure members to zero. */
```

```

memset(&pd, 0, sizeof(PRINTDLG));

/* Initialize the necessary PRINTDLG structure members. */

pd.lStructSize = sizeof(PRINTDLG);
pd.hwndOwner = hwnd;
pd.Flags = PD_RETURNDC;

/* Print a test page if successful */

if (PrintDlg(&pd) != 0) {
    Escape(pd.hDC, STARTDOC, 8, "Test-Doc", NULL);

    /* Print text and rectangle */

    TextOut(pd.hDC, 50, 50, "Common Dialog Test Page", 23);
    Rectangle(pd.hDC, 50, 90, 625, 105);
    Escape(pd.hDC, NEWFRAME, 0, NULL, NULL);
    Escape(pd.hDC, ENDDOC, 0, NULL, NULL);
    DeleteDC(pd.hDC);
    if (pd.hDevMode != NULL)
        GlobalFree(pd.hDevMode);
    if (pd.hDevNames != NULL)
        GlobalFree(pd.hDevNames);
}
else {
    if (pd.hDevMode != NULL)
        GlobalFree(pd.hDevMode);
    if (pd.hDevNames != NULL)
        GlobalFree(pd.hDevNames);
    ErrorHandler();
}

```

## Find and Replace Dialog Boxes (3.1)

### Using Find and Replace Dialog Boxes

The Find dialog box and the Replace dialog box are similar in appearance. You can use the Find dialog box to add string-search capabilities to your application and use the Replace dialog box to add both string-search and string-substitution capabilities.

### Displaying the Find Dialog Box

The Find dialog box contains controls that make it possible for a user to specify the following:

- The string that the application should find
- Whether the string specifies a complete word or part of a word
- Whether the application should match the case of the specified string
- The direction in which the application should search (preceding or following the current cursor location)
- Whether the application should resume the search, searching for the next occurrence of the string

Following is a Find dialog box.

- To display the Find dialog box, you need to initialize a FINDREPLACE structure and call the FindText function. Members of the FINDREPLACE structure contain information about such items as the following:

- Which window owns the dialog box
- How the application should perform the search
- A character buffer that is to receive the string

To initialize the FINDREPLACE structure, you need to perform the following tasks:

- 1 Set the lStructSize member by using the `sizeof` operator.
- 2 Set the hwndOwner member by using the handle that identifies the owner window of the dialog box.
- 3 If you are customizing the Find dialog box, set the hInstance member to identify the instance of the module that contains your custom dialog box template.
- 4 Set the Flags member to indicate the selection state of the dialog box options. (For example, setting the FR\_NOUPDOWN flag disables the Up and Down buttons, setting the FR\_NOWHOLEWORD flag disables the Match Whole Word Only check box, and setting the FR\_NOMATCHCASE flag disables the Match Case check box).
- 5 If you are supplying a custom dialog box template or hook function, set additional flags in the Flags member.
- 6 Set the lpstrFindWhat member to point to the buffer that will receive the string to be found.
- 7 Set the wFindWhatLen member to specify the size, in bytes, of the buffer to which lpstrFindWhat points.
- 8 Set the lCustData member with any custom data your application may need to access.
- 9 If your application customizes the Find dialog box, set the lpfnHook member to point to your hook function.
- 10 If your application uses a custom dialog box template, set the lpTemplateName member to point to the string that identifies the template.

The following example initializes the FINDREPLACE structure and then calls the FindText function. This structure should be global or declared as a static variable.

```
static FINDREPLACE fr;  
  
/* Set all structure fields to zero. */  
  
memset(&fr, 0, sizeof(FINDREPLACE));  
  
fr.lStructSize = sizeof(FINDREPLACE);  
fr(hwndOwner = hwnd;  
fr.lpstrFindWhat = szFindWhat;  
fr.wFindWhatLen = sizeof(szFindWhat);  
  
hDlg = FindText(&fr);
```

```
break;
```

### Displaying the Replace Dialog Box

The Replace dialog box is similar to the Find dialog box. However, the Replace dialog box has no Direction box and has three additional controls that make it possible for the user to specify the following:

- The replacement string
- Whether the application should replace the occurrence of the string that is currently highlighted
- Whether the application should replace all occurrences of the string

Following is a Replace dialog box.

To display the Replace dialog box, you need to initialize a FINDREPLACE structure and call the ReplaceText function.

### Processing Dialog Box Messages for a Find or Replace Dialog Box

The Find and Replace dialog boxes differ from the other common dialogs in two respects: First, they are modeless; and second, their respective dialog box procedures send messages to the application that calls the FindText or ReplaceText function. These messages contain data specified by the user in the dialog box controls, such as the direction in which the application should search for a string, whether the application should match the case of the specified string, and whether the application should match the entire string.

To process messages from a Find or Replace dialog box, an application must register the dialog box's unique message, FINDMSGSTRING.

The application registers this message with the RegisterWindowMessage function. Once the application registers the message, it uses the function's return value to identify messages from the Find or Replace dialog box. The following example registers the message with the RegisterWindowMessage function:

```
UINT uFindReplaceMsg;
```

```
/* Register the FindReplace message. */
```

```
uFindReplaceMsg = RegisterWindowMessage(FINDMSGSTRING);
```

After the application registers this message, it can process messages for the Find or Replace dialog box by using the RegisterWindowMessage return value. The following example processes messages for the Find dialog box and then calls its own SearchFile function to locate the string of text. If the user is closing the dialog box (that is, if the Flags member of the FINDREPLACE structure is FR\_DIALOGTERM), the handle is invalidated and the procedure returns zero.

```
LRESULT CALLBACK MainWndProc(HWND hwnd, UINT msg, WPARAM wParam,  
    LPARAM lParam)
```

```
{  
    static FINDREPLACE FAR* lpfr;  
  
    if (msg == uFindReplaceMsg) {  
        lpfr = (FINDREPLACE FAR*) lParam;  
        if (lpfr->Flags & FR_DIALOGTERM) {  
            hDlg = NULL;  
            return 0;  
        }  
        SearchFile((BOOL) (lpfr->Flags & FR_DOWN),  
            (BOOL) (lpfr->Flags & FR_MATCHCASE));  
        return 0;  
    }  
}
```

## Customizing Common Dialog Boxes (3.1)

A custom common dialog box is a common dialog box that has been altered to suit a particular Windows application. The customization may be complex and include the hiding of original controls, the addition of new controls, or a change in the size of the original dialog box; or it may be simple, such as the alteration of a single existing control.

Developers who need to customize a common dialog box must provide a special hook function and, in most cases, a custom dialog box template. Customizations of this kind require a significant amount of additional code--displaying a customized common dialog box is not as simple as initializing the members of a structure and calling a single function.

Applications that subclass controls in any of the common dialog boxes must do so while processing the `WM_INITDIALOG` message in the application's hook function. This allows the application to receive the control-specific messages first, because it will have subclassed the control after the common dialog box has installed its subclassing procedures. (The previous hook function should be called for all messages that are not handled by the application's subclass function, as is standard for subclassing.)

An application cannot subclass a control by defining a local class to override a specific control type. The reason is that the data segment would not be correctly initialized when the class was called--the data segment would be the common dialog box's data segment, not the application's data segment.

### Appropriate and Inappropriate Customizations

From the user's perspective, the chief benefit of the common dialog box is its consistent appearance and functionality from application to application. Therefore, it becomes important that a developer only customize a common dialog box when it is absolutely necessary for an application. Otherwise, the consistent appearance and simple coding interface are lost. Appropriate customizations leave intact as many of the original controls as possible. Increasing the size of the dialog box or adding new controls in available space that already appears in the dialog box would be an appropriate customization. Hiding original controls or otherwise changing the intended functionality of the original controls would be an inappropriate customization.

### Hook Functions and Custom Dialog Box Templates

Each common dialog box uses the dialog box procedure and dialog box template provided for it in `COMMDLG.DLL`. The dialog box procedure processes messages and notifications for the common dialog box and its controls. The dialog box template defines the appearance of the dialog box--its dimensions, its location, and the dimensions and locations of controls that appear within it.

In addition to the provided dialog box procedure and dialog box template, a custom dialog box requires a hook function that you provide and, usually, a custom version of the dialog box template.

### The Hook Function

The dialog box procedure provided in `COMMDLG.DLL` for a common dialog box calls the application's hook function if the application sets the appropriate flag and pointer in the structure for that common dialog box. The structure for each common dialog box contains a `Flags` member that specifies whether the application supplies a hook function and contains a `lpfnHook` member that points to the hook function if one exists. If the application sets the `Flags` member to indicate that a hook function exists, it must also set the `lpfnHook` member. The following example sets the `Flags` and `lpfnHook` members of an `OPENFILENAME` structure to support an application's hook function:

```
#define STRICT

#include <windows.h>      /* required for all Windows applications */
#include <commdlg.h>
#include <string.h>
#include "header.h" /* specific to this program */

OPENFILENAME ofn;

/* Get the system directory name, and store in szDirName. */

GetSystemDirectory((LPSTR)szDirName, 255);

/* Initialize the OPENFILENAME members. */
```



```

szFile[0] = '\\0';
ofn.lStructSize = sizeof(OPENFILENAME);
ofn.hwndOwner = hwnd;
ofn.hInstance = hInst;
ofn.lpstrFilter = szFilter[0];
ofn.lpstrCustomFilter = NULL;
ofn.nMaxCustFilter = 0L;
ofn.nFilterIndex = 1L;
ofn.lpstrFile = szFile;
ofn.nMaxFile = sizeof(szFile);
ofn.lpstrFileName = szFileName;
ofn.nMaxFileName = sizeof(szFileName);
ofn.lpstrInitialDir = szDirName;
ofn.lpstrTitle = NULL;
ofn.Flags = OFN_ENABLEHOOK | OFN_ENABLETEMPLATE;
ofn.nFileOffset = 0;
ofn.nFileExtension = 0;
ofn.lpstrDefExt = NULL;
ofn.lpfnHook = MakeProcInstance((FARPROC) FileOpenHookProc, hInst)
;
ofn.lpTemplateName = "FileOpen";

```

In the previous example, the MakeProcInstance function is called to create a procedure-instance address for the hook function. This address is assigned to the lpfnHook member of the OPENFILENAME structure. If the hook function is part of a dynamic-link library (rather than an application), the procedure address is obtained by calling the GetProcAddress function (instead of MakeProcInstance).

The hook function processes any messages or notifications that the custom dialog box requires. With the exception of one message (WM\_INITDIALOG), the hook function receives messages and notifications before the dialog box procedure provided in COMMDLG.DLL receives them. In the case of WM\_INITDIALOG, the hook function receives the message after the dialog box procedure. When the hook function finishes processing a message, it returns a value that indicates whether the dialog box procedure provided in COMMDLG.DLL should also process the message. If the dialog box procedure should process the message, the return value is FALSE; if the dialog box procedure should ignore the message, the return value is TRUE.

To process the message from the OK button after the dialog box procedure processes it, an application must post a message to itself when the OK message is received. When the application receives the message it has posted, the common dialog box procedure will have finished processing messages for the dialog box. This technique is particularly useful when working with the Find and Replace dialog boxes, because the Flags member of the FINDREPLACE structure does not reflect changes to the dialog box until after the messages have been processed by COMMDLG.DLL.

The following example shows a hook function for a custom Open dialog box:

```

UINT CALLBACK FileOpenHookProc(HWND hdlg, UINT msg, WPARAM wParam,
LPARAM lParam)
{
    switch(msg) {
        case WM_INITDIALOG:
            return TRUE;

        case WM_COMMAND:
            /* Use IsDlgButtonChecked to set lCustData. */

            if (wParam == IDOK) {
                /* Set backup flag. */

                ofn.lCustData =
                (DWORD) IsDlgButtonChecked(hdlg, ID_CUSTCHX);
            }

            return FALSE; /* Allow standard processing. */
    }
}

```

```

    /* Allow standard processing. */
    return FALSE;
}

```

This hook function tests a custom check box when the user chooses the OK button. If the check box was selected, the hook function sets the `ICustData` member of the `OPENFILENAME` structure to 1; otherwise, it sets the `ICustData` member to 0.

A hook function should never call the `EndDialog` function. Instead, if a hook function contains code that abnormally terminates a common dialog box, this code should pass the `IDABORT` value to the dialog box procedure by using the `PostMessage` function as shown in the following example:

```
PostMessage(hDlg, WM_COMMAND, IDABORT, (LONG) FALSE);
```

When a hook function posts the `IDABORT` value, the common dialog box function returns the value contained in the low word of the `lParam` parameter. For example, if the hook function for `GetOpenFileName` called the `PostMessage` function with `(LONG) 100` as the last parameter, `GetOpenFileName` would return 100.

A hook function must be exported in an application's module-definition (`.DEF`) file as shown in the following example:

```

NAME cd

EXETYPE WINDOWS

STUB 'WINSTUB.EXE'

CODE PRELOAD MOVEABLE DISCARDABLE

DATA PRELOAD MOVEABLE MULTIPLE

HEAPSIZE 1024

STACKSIZE 8192

EXPORTS
    FILEOPENHOOKPROC @1

```

### Customizing a Dialog Box Template

The dialog box template provided in `COMMDDL.DLL` for each common dialog box contains the data that the dialog box procedure uses to display that common dialog box. Most applications that customize a common dialog box also need to create a custom dialog box template to use instead of the dialog box template in `COMMDDL.DLL`. (A custom dialog box template is not required for all custom dialog boxes. For instance, a template would not be necessary if an application changed a dialog box in a relatively minor way and only in an unusual situation.)

A developer should create a custom dialog box template by modifying the appropriate dialog box template in `COMMDDL.DLL`. Following are the template filenames and the names of their corresponding common dialog boxes:

Template filename	Corresponding dialog box
<code>COLOR.DLG</code>	Color
<code>FILEOPEN.DLG</code>	Open (single selection)
<code>FILEOPEN.DLG</code>	Open (multiple selection)
<code>FINDTEXT.DLG</code>	Find
<code>FINDTEXT.DLG</code>	Replace
<code>FONT.DLG</code>	Font
<code>PRNSETUP.DLG</code>	Print
<code>PRNSETUP.DLG</code>	Print Setup

The following excerpt is from a custom dialog box template created for an Open dialog box:

```
CONTROL "&Backup File", ID_CUSTCHX, "button",
```

```
BS_AUTOCHECKBOX | WS_CHILD | WS_TABSTOP | WS_GROUP,  
208, 86, 50, 12
```

END

This entry supports the addition of a new Backup File check box immediately below the existing Read Only check box.

The custom template should be added to the application's resource file. You must use all of the unique control identifiers (that is, all identifiers whose values are other than -1) in the template, even if the dialog box does not use those controls. If you do not want to display all of the controls, you can specify coordinates for them that are outside the dialog box. You should also disable the unwanted buttons and remove unnecessary tab stops.

#### Displaying the Custom Dialog Box

After your application creates the hook function and the dialog box template, it should set the members of the structure for the common dialog box being customized and call the appropriate function to display the custom dialog box.

The following example calls the `GetOpenFileName` function and creates a backup file if the user selected the custom Backup File check box in the custom Open dialog box:

```
/* Open the file and create a backup. */  
  
if (GetOpenFileName(&ofn)) {  
    hf = lopen(ofn.lpstrFile, OF_READWRITE);  
    /* Create the backup file. */  
    if (ofn.lCustData) {  
        /* Process files with extension. */  
        if (ofn.nFileExtension){  
            for (i=0; i<(int)ofn.nFileExtension; i++)  
                szChar[i] = *ofn.lpstrFile++;  
            /*endif */  
            /* Process files without extension. */  
            else {  
                i=0;  
                while (*ofn.lpstrFile!='\0')  
                    szChar[i++] = *ofn.lpstrFile++;  
                szChar[i]='.';  
            }/*end else*/  
            pszNewPAFN = lstrcat(szChar, "BAK");  
            /* Create the backup file. */  
            hfBackup = lcreat(pszNewPAFN, 0);  
            /* Copy contents of original file to the backup file. */  
            while ((cBufLngth=lread(hf, cBuf1, 256)) == 256)  
                lwrite(hfBackup, cBuf1, cBufLngth);  
                lwrite(hfBackup, cBuf1, cBufLngth);  
                lclose(hfBackup);  
            }/*endif GetOpenFileName*/  
}
```

```

    /* File operations begin here. */
} /* endif (GetOpenFileName) */

```

The following is the custom Open dialog box. The new Backup File check box appears in the lower-right corner.

## Supporting and Processing Help for the Common Dialog Boxes

An application can display a Help button in any of the common dialog boxes by setting the appropriate flag in the Flags member of the structure for that common dialog box. Following are the structures for the common dialog boxes and the Help flag that corresponds to each structure:

Structure	Flag value
<u>OPENFILENAME</u>	OFN_SHOWHELP
<u>CHOOSECOLOR</u>	CC_SHOWHELP
<u>FINDREPLACE</u>	FR_SHOWHELP
<u>CHOOSEFONT</u>	CF_SHOWHELP
<u>PRINTDLG</u>	PD_SHOWHELP

If an application displays the Help button, it must process the user's request for Help. This can be done either in one of the application's window procedures or in a hook function.

If the application processes the request for Help in one of the application's window procedures, it must first create a new message identifier for the string defined by the HELPMMSGSTRING constant. The application creates the new message identifier by calling the RegisterWindowMessage function and passing this constant as the single parameter. (For more information about processing registered window messages, see Using Find and Replace Dialog Boxes.) In addition to creating a new message identifier, the application must set the hwndOwner member of the appropriate structure for the common dialog box so that this member contains the handle of the dialog box's owner window. After the message identifier is created and the hwndOwner member is set, the dialog box procedure notifies the window procedure of the owner window whenever the user chooses the Help button.

The following example processes a user's request for Help in the window procedure of its owner window. The if statement should be in the default: section of the switch statement that processes messages.

```

MyHelpMsg = RegisterWindowMessage(HELPMMSGSTRING);
.
.
.
if (message == MyHelpMsg)
WinHelp(hwnd, "appfile.hlp", HELP_CONTEXT, ID_MY_CONTEXT);

```

If the application processes the request for Help in a hook function, it should test for the following condition in the WM\_COMMAND message:

```
wParam == pshHelp
```

When this condition is true, the hook function should call the WinHelp function as shown in the preceding example. (To process Help in a hook function, you must include the header file DLGS.H in the source file that contains the hook-function code.)

### Error Detection

Whenever a common dialog box function fails, an application can call the CommDlgExtendedError function to find out the cause of the failure. The CommDlgExtendedError function returns an error value that identifies the cause of the most recent error.

Six constants are defined in the CDERR.H header file that identify the ranges of error values for

categories of errors returned by CommDlgExtendedError. Following are these constants in ascending order by value range:

Constant	Meaning
CDERR_GENERALCODES	General error codes for common dialog boxes. These errors are in the range 0x0000 through 0x0FFF.
PDERR_PRINTERCODES	Error codes for the Print common dialog box. These errors are in the range 0x1000 through 0x1FFF.
CFERR_CHOOSEFONTCODES	Error codes for the Font common dialog box. These errors are in the range 0x2000 through 0x2FFF.
FNERR_FILENAMECODES	Error codes for the Open and Save As common dialog boxes. These errors are in the range 0x3000 through 0x3FFF.
FRERR_FINDREPLACECODES	Error codes for the Find and Replace common dialog boxes. These errors are in the range 0x4000 through 0x4FFF.
CCERR_CHOOSECOLORCODES	Error codes for the Color common dialog box. These errors are in the range 0x5000 through 0x5FFF.

## Dynamic Data Exchange Management Library (3.1)

This topic describes how to use the Dynamic Data Exchange Management Library (DDEML). The DDEML is a dynamic-link library (DLL) that applications running with the Microsoft Windows operating system can use to share data.

The following topics discuss the concepts of DDE and describe how to use the DDE Management Library to add DDE functionality to an application:

- DDEML Concepts
- DDEML Initialization
- Callback Function
- String Management
- Conversation Management
- Data Management
- Transaction Management
- Server-Name Service
- Error Detection
- DDEML Monitor Applications

Dynamic data exchange (DDE) is a form of interprocess communication that uses shared memory to exchange data between applications. Applications can use DDE for one-time data transfers and for ongoing exchanges in which the applications send updates to one another as new data becomes available.

Dynamic data exchange differs from the clipboard data-transfer mechanism that is also part of the Windows operating system. One difference is that the clipboard is almost always used as a one-time response to a specific action by the user--such as choosing the Paste command from a menu. Although DDE may also be initiated by a user, it typically continues without the user's further involvement.

The DDEML provides an application programming interface (API) that simplifies the task of adding DDE capability to a Windows application. Instead of sending, posting, and processing DDE messages directly, an application uses the functions provided by the DDEML to manage DDE conversations. (A DDE conversation is the interaction between client and server applications.) The DDEML also provides a facility for managing the strings and data that are shared among DDE applications. Instead of using atoms and pointers to shared memory objects, DDE applications create and exchange string handles, which identify strings, and data handles, which identify global memory objects. DDEML provides a service that makes it possible for a server application to register the service names that it supports. The names are broadcast to other applications in the system, which can then use the names to connect to the server. The DDEML also ensures compatibility among DDE applications by forcing them to implement the DDE protocol in a consistent manner.

Existing applications that use the message-based DDE protocol are fully compatible with those that use the DDEML. That is, an application that uses message-based DDE can establish conversations and perform transactions with applications that use the DDEML. Because of the many advantages of the DDEML, new applications should use it rather than the DDE messages.

The DDEML can run on systems that have Microsoft Windows version 3.0 or later installed. The DDEML does not support real mode. To use the API elements of the DDE management library, you must include the DDEML.H header file in your source files, link with DDEML.LIB, and ensure that DDEML.DLL resides in the system's path.

## DDEML Concepts

### Basic Concepts

The concepts in this section are key to understanding DDE and the DDEML.

### Client and Server Interaction

Dynamic data exchange always takes place between a client application and a server application. The client initiates the exchange by establishing a conversation with the server so that it can send transactions to the server. (A transaction is a request for data or services.) The server responds to these transactions by providing data or services to the client. A server can have many clients at the same time, and a client can request data from multiple servers. Also, an application can be both a client and a server. A client terminates a conversation when it no longer needs a server's data or services.

For example, a graphics application might contain a bar graph that represents a corporation's quarterly profits, and the data for the bar graph might be contained in a spreadsheet application. To obtain the latest profit figures, the graphics application (the client) establishes a conversation with the spreadsheet application (the server). The graphics application then sends a transaction to the spreadsheet application, requesting the latest profit figures.

### Transactions and the DDE Callback Function

The DDEML notifies an application of DDE activity that affects the application by sending transactions to the application's DDE callback function. A transaction is similar to a message--it is a named constant accompanied by other parameters that contain additional information about the transaction.

The DDEML passes a transaction to an application-defined DDE callback function, which carries out the appropriate action depending on the type of the transaction. For example, when a client application attempts to establish a conversation with a server application, the client calls the `DdeConnect` function. This causes the DDEML to send an `XTYP_CONNECT` transaction to the server's DDE callback function. The callback function can allow the conversation by returning TRUE to the DDEML, or it can deny the conversation by returning FALSE.

For a detailed discussion of transactions, see Transaction Management.

### Service Names, Topic Names, and Item Names

A DDE server uses a three-level hierarchy--service name (called "application name" in previous DDE documentation), topic name, and item name--to uniquely identify a unit of data that the server can exchange during a conversation. A service name is a string that a server application responds to when a client attempts to establish a conversation with the server. A client must specify this service name to be able to establish a conversation with the server. Although a server can respond to many service names, most servers respond to only one name.

A topic name is a string that identifies a logical data context. For servers that operate on file-based documents, topic names are typically filenames; for other servers, they are other application-specific strings. A client must specify a topic name along with a server's service name when it attempts to establish a conversation with a server.

An item name is a string that identifies a unit of data that a server can pass to a client during a transaction. For example, an item name might identify an integer, a string, several paragraphs of text, or a bitmap.

To a client, these names are the keys that make it possible for the client to establish a conversation with a server and to receive data from the server.

### System Topic

The System topic provides a context for information that may be of general interest to any DDE client. Server applications are encouraged to support the System topic at all times. (The System topic is defined in the `DDEML` header file as `SZDDESYS_TOPIC`.)

To find out which servers are present and the kinds of information they can provide, a client can request a conversation on the System topic with the service name set to NULL when the client application starts. Such wildcard conversations should be kept to a minimum, because they are costly in terms of system performance.

For more information about initiating DDE conversations, see Conversation Management.

A server should support the following item names within the System topic and any other item names that may be useful to a client:

Item	Description
SZDDE_ITEM_ITEMLIST	A list of the items that are supported under a non-System topic. (This list may vary from moment to moment and from topic to topic.)
SZDDESYS_ITEM_FORMATS	A list of clipboard format numbers that the server can render. This list should be ordered with the most descriptive formats first. A server may not be able to render all items in all formats within this list. At a minimum, a server should support the CF_TEXT clipboard format for item names associated with the System topic.
SZDDESYS_ITEM_HELP	General help information.
SZDDESYS_ITEM_RTNMSG	Supporting detail for the most recently used <u>WM_DDE_ACK</u> message. This is useful when more than 8 bits of application-specific return data are required.
SZDDESYS_ITEM_STATUS	An indication of the current status of the server. Typically, this item supports only the CF_TEXT format and contains the Ready or Busy string.
SZDDESYS_ITEM_SYSITEMS	A list of the items supported under the System topic by this server.
SZDDESYS_ITEM_TOPICS	A list of the topics supported by the server at the current time. (This list may vary from moment to moment.)

These item names are string constants defined in the DDEML header files. To obtain string handles for these strings, an application must use the DDEML string-management functions, just as it would for any other string in a DDEML application. For more information about managing strings, see String Management.



## DDEML Initialization

The DDEML requires that Windows be running; otherwise, the system cannot load the DDEML dynamic-link library. Before calling any DDEML function, an application should call the `GetWinFlags` function, checking the return value for the `WF_PMODE` flag. If this flag is returned, the application can call DDEML functions.

Before calling any other DDEML function, an application must call the `DdeInitialize` function. The `DdeInitialize` function obtains an instance identifier for the application, registers the application's DDE callback function with the DDEML, and specifies the transaction filter flags for the callback function.

The DDEML uses instance identifiers so that it can support applications that allow multiple DDEML instances. Each instance of an application must pass its instance identifier as the `idInst` parameter to any other DDEML function that requires it. An application that uses multiple DDEML instances should assign a different DDE callback function to each instance. This makes it possible for the application to identify each instance within its callback function.

The purpose for multiple DDEML instances is to support DLLs using the DDEML. It is not recommended that an application have multiple DDE instances.

Transaction filters optimize system performance by preventing the DDEML from passing unwanted transactions to the application's DDE callback function. An application sets the transaction filters when it calls the `DdeInitialize` function. An application should specify a transaction filter flag for each type of transaction that it does not process in its callback function. An application can change its transaction filters with a subsequent call to the `DdeInitialize` function. For a complete list of transaction filter flags, see the description of the `DdeInitialize` function in the Microsoft Windows Programmer's Reference, Volume 2.

For more information about transactions, see [Transaction Management](#).

The following example shows how to initialize an application to use the DDEML:

```
DWORD idInst = 0L; /* instance identifier */
HANDLE hInst; /* instance handle */
FARPROC lpDdeProc; /* procedure instance address */

lpDdeProc = MakeProcInstance((FARPROC) DdeCallback, hInst);
if (DdeInitialize(&idInst, /* receives instance identifier*/
    (PFNCALLBACK) lpDdeProc, /* address of callback function*/
    CBF_FAIL_EXECUTES | /* filter XTYP_EXECUTE transactions */
    CBF_FAIL_POKES, 0L); /* filter XTYP_POKE transactions */
    return FALSE;
```

This example obtains a procedure-instance address for the callback function named `DdeCallback` and then passes the address to the DDEML. The `CBF_FAIL_EXECUTES` and `CBF_FAIL_POKES` filters prevent the DDEML from passing `XTYP_EXECUTE` or `XTYP_POKE` transactions to the callback function.

An application should call the `DdeUninitialize` function when it no longer needs to use the DDEML. This function terminates any conversations currently open for the application and frees the DDEML resources that the system allocated for the application.

The DDEML may have difficulty terminating a conversation. This occurs when the other partner in a conversation fails to terminate its end of the conversation. In this case, the system enters a modal loop while it waits for any conversations to be terminated. A system-defined timeout period is associated with this loop. If the timeout period expires before the conversations have been terminated, a message box appears that gives the user the choice of waiting for another timeout period (Retry), waiting indefinitely (Ignore), or exiting the modal loop (Abort). An application should call `DdeUninitialize` after it has become invisible to the user and after its message loop has terminated.

## DDEML Callback Function

An application that uses the DDEML must provide a callback function that processes the DDE events that affect the application. The DDEML notifies an application of such events by sending transactions to the application's DDE callback function. The transactions that a callback function receives depend on the callback-filter flags that the application specified in the `DdeInitialize` function and on whether the application is a client, a server, or both. The following example shows the general structure of a callback function for a typical client application:

```
HDDEDATA CALLBACK DdeCallback(<type>, <fmt>, <hconv>, <hsz1>,
    <hsz2>, <hData>, <dwData1>, <dwData2>)
UINT <type>, /* transaction type */
UINT <fmt>, /* clipboard data format */
HCONV <hconv>, /* handle of conversation */
HSZ <hsz1>, /* handle of string */
HSZ <hsz2>, /* handle of string */
HDDEDATA <hData>, /* handle of global memory object */
DWORD <dwData1>, /* transaction-specific data */
DWORD <dwData2>, /* transaction-specific data */
{
    switch (type) {
        case XTYP_REGISTER:
        case XTYP_UNREGISTER:
            .
            .
            .
            return (HDDEDATA) NULL;

        case XTYP_ADVDATA:
            .
            .
            .
            return (HDDEDATA) DDE_FACK;

        case XTYP_XACT_COMPLETE:
            .
            .
            .
            return (HDDEDATA) NULL;

        case XTYP_DISCONNECT:
            .
            .
            .
            return (HDDEDATA) NULL;

        default:
            return (HDDEDATA) NULL;
    }
}
```

The type parameter specifies the transaction type sent to the callback function by the DDEML. The values of the remaining parameters depend on the transaction type. The transaction types and the events that generate them are described in the following sections of this topic. For detailed information about each transaction type, see Transaction Management.

## DDEML String Management

Many DDEML functions require access to strings in order to carry out a DDE task. For example, a client must specify a service name and a topic name when it calls the DdeConnect function to request a conversation with a server. An application specifies a string by passing a string handle rather than a pointer in a DDEML function. A string handle is a doubleword value, assigned by the system, that identifies a string.

An application can obtain a string handle for a particular string by calling the DdeCreateStringHandle function. This function registers the string with the system and returns a string handle to the application. The application can pass the handle to DDEML functions that need to access the string. The following example obtains string handles for the System topic string and the service-name string:

```
HSZ hszServName;
HSZ hszSysTopic;

hszServName = DdeCreateStringHandle(
    idInst, /* instance identifier */
    "MyServer", /* string to register */
    CP_WINANSI); /* code page */

hszSysTopic = DdeCreateStringHandle(
    idInst, /* instance identifier */
    SZDDSYS_TOPIC, /* System topic */
    CP_WINANSI); /* code page */
```

The idInst parameter in the preceding example specifies the instance identifier obtained by the DdeInitialize function.

An application's DDE callback function receives one or more string handles during most DDE transactions. For example, a server receives two string handles during the XTYP\_REQUEST transaction: One identifies a string specifying a topic name; the other identifies a string specifying an item name. An application can obtain the length of the string that corresponds to a string handle and copy the string to an application-defined buffer by calling the DdeQueryString function, as the following example demonstrates:

```
DWORD idInst;
DWORD cb;
HSZ hszServ;
PSTR pszServName;

cb = DdeQueryString(idInst, hszServ, (LPSTR) NULL, 0L, CP_WINANSI) + 1;
pszServName = (PSTR) LocalAlloc(LPTR, (WORD) cb);
DdeQueryString(idInst, hszServ, pszServName, cb, CP_WINANSI);
```

An instance-specific string handle is not mappable from string handle to string to string handle again. For instance, in the following example, the DdeQueryString function creates a string from a string handle and then DdeCreateStringHandle creates a string handle from that string, but the two handles are not the same:

```
DWORD cb;
HSZ hszInst, hszNew;
PSZ pszInst;

DdeQueryString(idInst, hszInst, pszInst, cb, CP_WINANSI);
hszNew = DdeCreateStringHandle(idInst, pszInst, CP_WINANSI);
/* hszNew != hszInst ! */
```

A string handle that is passed to an application's DDE callback function becomes invalid when the callback function returns. An application can save a string handle for use after the callback function returns by using the DdeKeepStringHandle function.

When an application calls DdeCreateStringHandle, the system enters the specified string into a systemwide string table and generates a handle that it uses to access the string. The system also maintains a usage count for each string in the string table.

When an application calls the DdeCreateStringHandle function and specifies a string that already exists in the table, the system increments the usage count rather than adding another occurrence of the string. (An

application can also increment the usage count by using the DdeKeepStringHandle function.) When an application calls the DdeFreeStringHandle function, the system decrements the usage count.

A string is removed from the table when its usage count equals zero. Because more than one application can obtain the handle of a particular string, an application should not free a string handle more times than it has created or kept the handle. Otherwise, the application could cause the string to be removed from the table, denying other applications access to the string.

The DDEML string-management functions are based on the Windows atom manager and are subject to the same size restrictions as atoms.

## DDEML Server Name Service

The DDEML makes it possible for a server application to register the service names that it supports and to prevent the DDEML from sending XTYP\_CONNECT transactions for unsupported service names to the server's DDE callback function. The remaining topics in this section describe this service.

### Service-Name Registration

By registering its service names with the DDEML, a server informs other DDE applications in the system that a new server is available. A server registers a service name by calling the DdeNameService function, specifying a string handle that identifies the name. As a result, the DDEML sends an XTYP\_REGISTER transaction to the callback function of each DDEML application in the system (except those that specified the CBF\_SKIP\_REGISTRATIONS filter flag in the DdeInitialize function). The XTYP\_REGISTER transaction passes two string handles to a callback function: The first identifies the string specifying the base service name; the second identifies the string specifying the instance-specific service. A client typically uses the base service name in a list of available servers, so that the user can select a server from the list. The client uses the instance-specific service name to establish a conversation with a specific instance of a server application if more than one instance is running.

A server can use the DdeNameService function to unregister a service name. This causes the DDEML to send XTYP\_UNREGISTER transactions to the other DDE applications in the system, informing them that they can no longer use the name to establish conversations.

A server should call the DdeNameService function to register its service names soon after calling the DdeInitialize function. A server should unregister its service names just before calling the DdeUninitialize function.

### Service-Name Filter

Besides registering service names, the DdeNameService function makes it possible for a server to turn its service-name filter on or off. When a server turns off its service-name filter, the DDEML sends the XTYP\_CONNECT transaction to the server's DDE callback function whenever any client calls the DdeConnect function, regardless of the service name specified in the function. When a server turns on its service-name filter, the DDEML sends the XTYP\_CONNECT transaction to the server only when the DdeConnect function specifies a service name that the server has specified in a call to the DdeNameService function.

By default, the service-name filter is on when an application calls the DdeInitialize function. This prevents the DDEML from sending the XTYP\_CONNECT transaction to a server before the server has created the string handles that it needs. A server can turn off its service-name filter by specifying the DNS\_FILTEROFF flag in a call to the DdeNameService function. The DNS\_FILTERON flag turns on the filter.

## DDEML Conversation Management

A conversation between a client and a server is always established at the request of the client. When a conversation is established, each partner receives a handle that identifies the conversation. The partners use this handle in other DDEML functions to send transactions and manage the conversation.

A client can request a conversation with a single server, or it can request multiple conversations with one or more servers. The remaining topics in this section describe how an application establishes conversations and explain how an application can obtain information about conversations that are already established.

### Single Conversations

A client application requests a single conversation with a server by calling the DdeConnect function, specifying string handles that identify the strings specifying the service name of the server and the topic name of interest. The DDEML responds by sending the XTYP\_CONNECT transaction to the DDE callback function of each server application that either has registered a service name that matches the one specified in the DdeConnect function or has turned service-name filtering off by calling the DdeNameService function. A server can also filter the XTYP\_CONNECT transactions by specifying the CBF\_FAIL\_CONNECTIONS filter flag in the DdeInitialize function. During the XTYP\_CONNECT transaction, the DDEML passes the service name and the topic name to the server. The server should examine the names and return TRUE if it supports the service/topic name pair or FALSE if it does not.

If no server returns TRUE from the XTYP\_CONNECT transaction, the client receives NULL from the DdeConnect function and no conversation is established. If a server does return TRUE, a conversation is established and the client receives a conversation handle--a doubleword value that identifies the conversation. The client uses the handle in subsequent DDEML calls to obtain data from the server. The server receives the XTYP\_CONNECT\_CONFIRM transaction (unless the server specified the CBF\_FAIL\_CONFIRMS filter flag). This transaction passes the conversation handle to the server.

The following example requests a conversation on the System topic with a server that recognizes the service name MyServer. The hszServName and hszSysTopic parameters are previously created string handles.

```
HCONV hConv;
HWND hwndParent;
HSZ hszServName;
HSZ hszSysTopic;

hConv = DdeConnect(
    idInst, /* instance identifier */
    hszServName, /* service-name string handle */
    hszSysTopic, /* System-topic string handle */
    (PCONVCONTEXT) NULL); /* reserved--must be NULL*/

if (hConv == NULL) {
    MessageBox(hwndParent, "MyServer is unavailable.",
        (LPSTR) NULL, MB_OK);
    return FALSE;
}
```

The DdeConnect function in the preceding example causes the DDE callback function of the MyServer application to receive an XTYP\_CONNECT transaction.

In the following example, the server responds to the XTYP\_CONNECT transaction by comparing the topic-name string handle that the DDEML passed to the server with each element in the array of topic-name string handles that the server supports. If the server finds a match, it establishes the conversation.

```
#define CTOPICS 5

HSZ hsz1; /* string handle passed by DDEML */
HSZ ahszTopics[CTOPICS]; /* array of supported topics */
int i; /* loop counter */

.
. /* Use switch statement to examine transaction types. */
.
```

```

case XTYP_CONNECT:
    for (i = 0; i < CTOPICS; i++) {
        if (hsz1 == ahszTopics[i])
            return TRUE; /* establish a conversation */
    }

    return FALSE; /* topic not supported; deny conversation */

.
. /* Process other transaction types. */
.

```

If the server returns TRUE in response to the XTYP\_CONNECT transaction, the DDEML sends an XTYP\_CONNECT\_CONFIRM transaction to the server's DDE callback function. The server can obtain the handle for the conversation by processing this transaction.

A client can establish a wildcard conversation by specifying NULL for the service-name string handle, the topic-name string handle, or both in a call to the DdeConnect function. When at least one of the string handles is NULL, the DDEML sends the XTYP\_WILDCONNECT transaction to the callback functions of all DDE applications (except those that filter the XTYP\_WILDCONNECT transaction). Each server application should respond by returning a data handle that identifies a null-terminated array of HSZPAIR structures. If the server application has not called the DdeNameService function to register its service names and filtering is on, the server does not receive XTYP\_WILDCONNECT transactions. For more information about data handles, see Data Management.

The array should contain one structure for each service/topic name pair that matches the pair specified by the client. The DDEML selects one of the pairs to establish a conversation and returns to the client a handle that identifies the conversation. The DDEML sends the XTYP\_CONNECT\_CONFIRM transaction to the server (unless the server filters this transaction). The following example shows a typical server response to the XTYP\_WILDCONNECT transaction:

```

#define CTOPICS 2

UINT type;
UINT fmt;
HSZPAIR ahp[(CTOPICS + 1)];
HSZ ahszTopicList[CTOPICS];
HSZ hszServ, hszTopic;
WORD i, j;

if (type == XTYP_WILDCONNECT) {

/*
 * Scan the topic list, and create array of HSZPAIR
 * structures.
 */

    j = 0;
    for (i = 0; i < CTOPICS; i++) {
        if (hszTopic == (HSZ) NULL ||
            hszTopic == ahszTopicList[i]) {
            ahp[j].hszSvc = hszServ;
            ahp[j++].hszTopic = ahszTopicList[i];
        }
    }

    /*
 * End the list with an HSZPAIR structure that contains NULL
 * string handles as its members.
 */

    ahp[j].hszSvc = NULL;
    ahp[j++].hszTopic = NULL;

    /*
 * Return a handle to a global memory object containing the

```

```
* HSZPAIR structures.  
*/
```

```
    return DdeCreateDataHandle(  
    idInst, /* instance identifier */  
    &ahp, /* points to HSZPAIR array */  
    sizeof(HSZ) * j, /* length of the array */  
    0, /* start at the beginning */  
    NULL, /* no item-name string */  
    fmt, /* return the same format */  
    0); /* let the system own it */  
}
```

Either the client or the server can terminate a conversation at any time by calling the DdeDisconnect function. This causes the callback function of the partner in the conversation to receive the XTYP\_DISCONNECT transaction (unless the partner specified the CBF\_SKIP\_DISCONNECTS filter flag). Typically, an application responds to the XTYP\_DISCONNECT transaction by using the DdeQueryConvInfo function to obtain information about the conversation that terminated. After the callback function returns from processing the XTYP\_DISCONNECT transaction, the conversation handle is no longer valid.

A client application that receives an XTYP\_DISCONNECT transaction in its DDE callback function can attempt to reestablish the conversation by calling the DdeReconnect function. The client must call DdeReconnect from within its DDE callback function.

#### Multiple Conversations

A client application can use the DdeConnectList function to determine whether any servers of interest are available in the system. A client specifies a service name and topic name when it calls the DdeConnectList function, causing the DDEML to broadcast the XTYP\_WILDCONNECT transaction to the DDE callback functions of all servers that match the service name (except those that filter the transaction). A server's callback function should return a data handle that identifies a null-terminated array of HSZPAIR structures. The array should contain one structure for each service/topic name pair that matches the pair specified by the client. The DDEML establishes a conversation for each HSZPAIR structure filled by the server and returns a conversation-list handle to the client. The server receives the conversation handle by way of the XTYP\_CONNECT\_CONFIRM transaction (unless the server filters this transaction).

A client can specify NULL for the service name, topic name, or both when it calls the DdeConnectList function. If the service name is NULL, all servers in the system that support the specified topic name respond. A conversation is established with each responding server, including multiple instances of the same server. If the topic name is NULL, a conversation is established on each topic recognized by each server that matches the service name.

A client can use the DdeQueryNextServer and DdeQueryConvInfo functions to identify the servers that respond to the DdeConnectList function. The DdeQueryNextServer function returns the next conversation handle in a conversation list; the DdeQueryConvInfo function fills a CONVINFO structure with information about the conversation. The client can keep the conversation handles that it needs and discard the rest from the conversation list.

The following example uses the DdeConnectList function to establish conversations with all servers that support the System topic and then uses the DdeQueryNextServer and DdeQueryConvInfo functions to obtain the servers' service-name string handles and store them in a buffer:

```
HCONVLIST hconvList; /* conversation list */  
DWORD idInst; /* instance identifier */  
HSZ hszSystem; /* System topic */  
HCONV hconv = NULL; /* conversation handle */  
CONVINFO ci; /* holds conversation data */  
UINT cConv = 0; /* count of conv. handles */  
HSZ *pHsz, *aHsz; /* point to string handles */  
  
/* Connect to all servers that support the System topic. */  
  
hconvList = DdeConnectList(idInst, NULL, hszSystem, NULL, NULL);  
  
/* Count the number of handles in the conversation list. */
```



```

while ((hconv = DdeQueryNextServer(hconvList, hconv)) != NULL) cConv++
;

/* Allocate a buffer for the string handles. */

hconv = NULL;
aHsz = (HSZ *) LocalAlloc(LMEM_FIXED, cConv * sizeof(HSZ));

/* Copy the string handles to the buffer. */

pHsz = aHsz;
while ((hconv = DdeQueryNextServer(hconvList, hconv)) != NULL) {
    DdeQueryConvInfo(hconv, QID_SYNC, (PCONVINFO) &ci);
    DdeKeepStringHandle(idInst, ci.hszSvcPartner);
    *pHsz++ = ci.hszSvcPartner;
}

.
. /* Use the handles; converse with servers. */
.

/* Free the memory, and terminate conversations. */

LocalFree((HANDLE) aHsz);
DdeDisconnectList(hconvList);

```

An application can terminate an individual conversation in a conversation list by calling the DdeDisconnect function. An application can terminate all conversations in a conversation list by calling the DdeDisconnectList function. Both functions cause the DDEML to send XTYP\_DISCONNECT transactions to each partner's DDE callback function. The DdeDisconnectList function sends a XTYP\_DISCONNECT transaction for each conversation handle in the list.

A client can use the DdeConnectList function to enumerate the conversation handles in a conversation list by passing an existing conversation-list handle to the DdeConnectList function. The enumeration process removes the handles of terminated conversations from the list.

If the DdeConnectList function specifies an existing conversation-list handle and a service name or topic name that is different from those used to create the existing conversation list, the function creates a new conversation list that contains the handles of any new conversations and the handles from the existing list.

The DdeConnectList function attempts to prevent duplicate conversations in a conversation list. A duplicate conversation is a second conversation with the same server on the same service name and topic name. Two such conversations would have different handles, yet they would be duplicate conversations.

## DDEML Data Management

Because DDE uses global memory to pass data from one application to another, the DDEML provides a set of functions that DDE applications can use to create and manage global memory objects.

All transactions that involve the exchange of data require the application supplying the data to create a local buffer containing the data and then to call the DdeCreateDataHandle function. This function allocates a global memory object, copies the data from the buffer to the memory object, and returns a data handle of the application. A data handle is a doubleword value that the DDEML uses to provide access to data in the global memory object. To share the data in a global memory object, an application passes the data handle to the DDEML, and the DDEML passes the handle to the DDE callback function of the application that is receiving the data transaction.

The following example shows how to create a global memory object and obtain a handle of the object. During the XTYP\_ADVREQ transaction, the callback function converts the current time to an ASCII string, copies the string to a local buffer, then creates a global memory object that contains the string. The callback function returns the handle of the global memory object to the DDEML, which passes the handle to the client application.

```
typedef struct { /* tm */
    int hour;
    int minute;
    int second;
} TIME;

TIME tmTime;
HSZ hszTime;
HSZ hszNow;
HDDEDATA EXPENTRY DdeProc(wType, wFmt, hConv, hsz1, hsz2,
    hData, dwData1, dwData2)
WORD wType;
WORD wFmt;
HCONV hConv;
HSZ hsz1;
HSZ hsz2;
HDDEDATA hData;
DWORD dwData1;
DWORD dwData2;
{
    char szBuf[32];

    switch (wType) {

        case XTYP_ADVREQ:
            if ((hsz1 == hszTime && hsz2 == hszNow)
                && (wFmt == CF_TEXT)) {

                /* Copy formatted time string to buffer. */

                itoa(tmTime.hour, szBuf, 10);
                strcat(szBuf, ":");
                if (tmTime.minute < 10)
                    strcat(szBuf, "0");
                itoa(tmTime.minute, &szBuf[strlen(szBuf)], 10);
                strcat(szBuf, ":");
                if (tmTime.second < 10)
                    strcat(szBuf, "0");
                itoa(tmTime.second, &szBuf[strlen(szBuf)], 10);
                szBuf[strlen(szBuf)] = '\0';

                /* Create global object, and return data handle. */

                return (DdeCreateDataHandle(
                    idInst, /* instance identifier */
                    (LPBYTE) szBuf, /* source buffer */
                    */*
```

```

strlen(szBuf) + 1, /* size of global object */
0L, /* offset from beginning */
hszNow, /* item-name string */
CF_TEXT, /* clipboard format */
0)); /* no creation flags*/
    } else
        return (HDDEDATA) NULL;

    .
    . /* Process other transaction types. */
    .
}
}

```

The receiving application obtains a pointer to the global memory object by passing the data handle to the DdeAccessData function. The pointer returned by DdeAccessData provides read-only access. The application should use the pointer to review the data and then call the DdeUnaccessData function to invalidate the pointer. The application can copy the data to a local buffer by using the DdeGetData function.

The following example obtains a pointer to the global memory object identified by the `hData` parameter, copies the contents to a local buffer, and then invalidates the pointer:

```

HDDEDATA hData;
LPBYTE lpszAdviseData;
DWORD cbDataLen;
DWORD i;
char szData[32];

case XTYP_ADVDATA:

    lpszAdviseData = DdeAccessData(hData, &cbDataLen);
    for (i = 0; i < cbDataLen; i++)
        szData[i] = *lpszAdviseData++;
    DdeUnaccessData(hData);
    return (HDDEDATA) TRUE;

```

Usually, when an application that created a data handle passes that handle to the DDEML, the handle becomes invalid in the creating application. This is fine if the application needs to share data with just a single application. If an application needs to share the same data with multiple applications, however, the creating application should specify the `HDATA_APPOWNED` flag in DdeCreateDataHandle. Doing so gives ownership of the memory object to the creating application and prevents the DDEML from invalidating the data handle. When the creating application finishes using a memory object it owns, it should free the object by calling the DdeFreeDataHandle function.

If an application has not yet passed the handle of a global memory object to the DDEML, the application can add data to the object or overwrite data in the object by using the DdeAddData function. Typically, an application uses DdeAddData to fill an uninitialized global memory object. After an application passes a data handle to the DDEML, the global memory object identified by the handle cannot be changed; it can only be freed.

The DDEML data-management functions can handle huge memory objects. A DDEML application should check the size of a global memory object and allocate a huge buffer of the appropriate size before copying the object.

## DDEML Transaction Management

After a client has established a conversation with a server, the client can send transactions to obtain data and services from the server. The remaining topics in this section describe the types of transactions that clients can use to interact with a server.

### Request Transaction

A client application can use the XTYP\_REQUEST transaction to request a data item from a server application. The client calls the DdeClientTransaction function, specifying XTYP\_REQUEST as the transaction type and specifying the data item the application needs.

The DDEML passes the XTYP\_REQUEST transaction to the server, specifying the topic name, item name, and data format requested by the client. If the server supports the requested topic, item, and data format, the server should return a data handle that identifies the current value of the item. The DDEML passes this handle to the client as the return value from the DdeClientTransaction function. The server should return NULL if it does not support the topic, item, or format requested.

The DdeClientTransaction function uses the lpdwResult parameter to return a transaction status flag to the client. If the server does not process the XTYP\_REQUEST transaction, DdeClientTransaction returns NULL, and lpdwResult points to the DDE\_FNOTPROCESSED or DDE\_FBUSY flag. If the DDE\_FNOTPROCESSED flag is returned, the client has no way to determine why the server did not process the transaction.

If a server does not support the XTYP\_REQUEST transaction, it should specify the CBF\_FAIL\_REQUESTS filter flag in the DdeInitialize function. This prevents the DDEML from sending this transaction to the server.

### Poke Transaction

A client can send unsolicited data to a server by using the DdeClientTransaction function to send an XTYP\_POKE transaction to a server's callback function.

The client application first creates a buffer that contains the data to send to the server and then passes a pointer to the buffer as a parameter to the DdeClientTransaction function. Alternatively, the client can use the DdeCreateDataHandle function to obtain a data handle that identifies the data and then pass the handle to DdeClientTransaction. In either case, the client also specifies the topic name, item name, and data format when it calls DdeClientTransaction.

The DDEML passes the XTYP\_POKE transaction to the server, specifying the topic name, item name, and data format that the client requested. To accept the data item and format, the server should return DDE\_FACK. To reject the data, the server should return DDE\_FNOTPROCESSED. If the server is too busy to accept the data, the server should return DDE\_FBUSY.

When the DdeClientTransaction function returns, the client can use the lpdwResult parameter to access the transaction status flag. If the flag is DDE\_FBUSY, the client should send the transaction again later.

If a server does not support the XTYP\_POKE transaction, it should specify the CBF\_FAIL\_POKES filter flag in the DdeInitialize function. This prevents the DDEML from sending this transaction to the server.

### Advise Transaction

A client application can use the DDEML to establish one or more links to items in a server application. When such a link is established, the server sends periodic updates about the linked item to the client (typically, whenever the value of the item associated with the server application changes). This establishes an advise loop between the two applications that remains in place until the client ends it.

There are two kinds of advise loops: "hot" and "warm." In a hot advise loop, the server immediately sends a data handle that identifies the changed value. In a warm advise loop, the server notifies the client that the value of the item has changed but does not send the data handle until the client requests it.

A client can request a hot advise loop with a server by specifying the XTYP\_ADVSTART transaction type in a call to the DdeClientTransaction function. To request a warm advise loop, the client must combine the XTYPF\_NODATA flag with the XTYP\_ADVSTART transaction type. In either event, the DDEML passes the XTYP\_ADVSTART transaction to the server's DDE callback function. The server's DDE callback function should examine the parameters that accompany the XTYP\_ADVSTART transaction (including the requested format, topic name, and item name) and then return TRUE to allow the advise loop or FALSE to deny it.

After an advise loop is established, the server application should call the DdePostAdvise function whenever the value of the item associated with the requested item name changes. This results in an

XTYP\_ADVREQ transaction being sent to the server's own DDE callback function. The server's DDE callback function should return a data handle that identifies the new value of the data item. The DDEML then notifies the client that the specified item has changed by sending the XTYP\_ADVDATA transaction to the client's DDE callback function.

If the client requested a hot advise loop, the DDEML passes the data handle for the changed item to the client during the XTYP\_ADVDATA transaction. Otherwise, the client can send an XTYP\_REQUEST transaction to obtain the data handle.

It is possible for a server to send updates faster than a client can process the new data. This can be a problem for a client that must perform long processing operations on the data. In this case, the client should specify the XTYPF\_ACKREQ flag when it requests an advise loop. This causes the server to wait for the client to acknowledge that it has received and processed a data item before the server sends the next data item. Advise loops that are established with the XTYPF\_ACKREQ flag are more robust with fast servers but may occasionally miss updates. Advise loops established without the XTYPF\_ACKREQ flag are guaranteed not to miss updates as long as the client keeps up with the server.

A client can end an advise loop by specifying the XTYP\_ADVSTOP transaction type in a call to the DdeClientTransaction function.

If a server does not support advise loops, it should specify the CBF\_FAIL\_ADVISES filter flag in the DdeInitialize function. This prevents the DDEML from sending the XTYP\_ADVSTART and XTYP\_ADVSTOP transactions to the server.

#### Execute Transaction

A client can use the XTYP\_EXECUTE transaction to cause a server to execute a command or series of commands.

To execute a server command, the client first creates a buffer that contains a command string for the server to execute and then passes either a pointer to the buffer or a data handle identifying the buffer when it calls the DdeClientTransaction function. Other required parameters include the conversation handle, the item-name string handle, the format specification, and the XTYP\_EXECUTE transaction type. When an application creates a data handle for passing execute data, the application must specify NULL for the hszItem parameter of the DdeCreateDataHandle function.

The DDEML passes the XTYP\_EXECUTE transaction to the server's DDE callback function specifying the format name, conversation handle, topic name, and data handle identifying the command string. If the server supports the command, it should use the DdeAccessData function to obtain a pointer to the command string, execute the command, and then return DDE\_FACK. If the server does not support the command or cannot complete the transaction, it should return DDE\_FNOTPROCESSED. The server should return DDE\_FBUSY if it is too busy to complete the transaction.

When the DdeClientTransaction function returns, the client can use the lpdwResult parameter to access the transaction status flag. If the flag is DDE\_FBUSY, the client should send the transaction again later.

If a server does not support the XTYP\_EXECUTE transaction, it should specify the CBF\_FAIL\_EXECUTE filter flag in the DdeInitialize function. Doing so prevents the DDEML from sending this transaction to the server.

#### Synchronous and Asynchronous Transactions

A client can send either synchronous or asynchronous transactions. In a synchronous transaction, the client specifies a timeout value that indicates the maximum amount of time to wait for the server to process the transaction. The DdeClientTransaction function does not return until the server processes the transaction, the transaction fails, or the timeout value expires. The client specifies the timeout value when it calls DdeClientTransaction.

During a synchronous transaction, the client enters a modal loop while waiting for the transaction to be processed. The client can still process user input but cannot send another synchronous transaction until the DdeClientTransaction function returns.

A client sends an asynchronous transaction by specifying the TIMEOUT\_ASYNC flag in the DdeClientTransaction function. The function returns after the transaction is begun, passing a transaction identifier to the client. When the server finishes processing the asynchronous transaction, the DDEML sends an XTYP\_XACT\_COMPLETE transaction to the client. One of the parameters that the DDEML passes to the client during the XTYP\_XACT\_COMPLETE transaction is the transaction identifier. By comparing this transaction identifier with the identifier returned by the DdeClientTransaction function, the client identifies which asynchronous transaction the server has finished processing.

A client can use the DdeSetUserHandle function as an aid to processing an asynchronous transaction. This function makes it possible for a client to associate an application-defined doubleword value with a

conversation handle and transaction identifier. The client can use the DdeQueryConvInfo function during the XTYP\_XACT\_COMPLETE transaction to obtain the application-defined doubleword value. This saves an application from having to maintain a list of active transaction identifiers.

If a server does not process an asynchronous transaction in a timely manner, the client can abandon the transaction by calling the DdeAbandonTransaction function. The DDEML releases all resources associated with the transaction and discards the results of the transaction when the server finishes processing it.

The asynchronous transaction method is provided for applications that must send a high volume of DDE transactions while simultaneously performing a substantial amount of processing, such as calculations. The asynchronous method is also useful in applications that need to stop processing DDE transactions temporarily so they can complete other tasks without interruption. In most other situations, an application should use the synchronous method.

Synchronous transactions are simpler to maintain and faster than asynchronous transactions. However, only one synchronous transaction can be performed at a time, whereas many asynchronous transactions can be performed simultaneously. With synchronous transactions, a slow server can cause a client to remain idle while waiting for a response. Also, synchronous transactions cause the client to enter a modal loop that could bypass message filtering in the application's own message loop.

#### Transaction Control

An application can suspend transactions to its DDE callback function--either those transactions associated with a specific conversation handle or all transactions regardless of the conversation handle. This is useful when an application receives a transaction that requires lengthy processing. In this case, an application can return CBR\_BLOCK to suspend future transactions associated with that transaction's conversation handle, leaving the application free to process other conversations.

When processing is complete, the application calls the DdeEnableCallback function to resume transactions associated with the suspended conversation. Calling DdeEnableCallback causes the DDEML to resend the transaction that resulted in the application suspending the conversation. Therefore, the application should store the result of the transaction in such a way that it can obtain and return the result without reprocessing the transaction.

An application can suspend all transactions associated with a specific conversation handle by specifying the handle and the EC\_DISABLE flag in a call to the DdeEnableCallback function. By specifying a NULL handle, an application can suspend all transactions for all conversations.

When a conversation is suspended, the DDEML saves transactions for the conversation in a transaction queue. When the application reenables the conversation, the DDEML removes the saved transactions from the queue, passing each transaction to the appropriate callback function. Even though the capacity of the transaction queue is large, an application should reenable a suspended conversation as soon as possible to avoid losing transactions.

An application can resume usual transaction processing by specifying the EC\_ENABLEALL flag in the DdeEnableCallback function. For a more controlled resumption of transaction processing, the application can specify the EC\_ENABLEONE flag. This removes one transaction from the transaction queue and passes it to the appropriate callback function; after the single transaction is processed, any conversations are again disabled.

#### Transaction Classes

The DDEML has four classes of transactions. Each class is identified by a constant that begins with the XCLASS\_ prefix. The classes are defined in the DDEML header file. The class constant is combined with the transaction-type constant and is passed to the DDE callback function of the receiving application.

A transaction's class determines the return value that a callback function is expected to return if it processes the transaction. The following table shows the return values and transaction types associated with each of the four transaction classes:

Class	Return value	Transaction
<u>XCLASS_BOOL</u>	TRUE or FALSE	<u>XTYP_ADVSTART</u> <u>XTYP_CONNECT</u>
<u>XCLASS_DATA</u>	A data handle, <u>CBR_BLOCK</u> , or <u>NULL</u>	<u>XTYP_ADVREQ</u> <u>XTYP_REQUEST</u> <u>XTYP_WILDCONNECT</u>
<u>XCLASS_FLAGS</u>	A transaction flag: <u>DDE_FACK</u> , <u>DDE_FBUSY</u> , or <u>DDE_FNOTPROCESSED</u>	<u>XTYP_ADVDATA</u> <u>XTYP_EXECUTE</u> <u>XTYP_POKE</u>

XCLASS_NOTIFICATION	None	XTYP_ADVSTOP XTYP_CONNECT_CONFIRM XTYP_DISCONNECT XTYP_ERROR XTYP_REGISTER XTYP_UNREGISTER XTYP_XACT_COMPLETE
---------------------	------	--

Transaction Summary

The following list shows each DDE transaction type, the receiver of each type, and a description of the activity that causes the DDEML to generate each type:

Transaction type	Receiver	Cause
XTYP_ADVDATA	Client	A server responded to an XTYP_ADVREQ transaction by returning a data handle.
XTYP_ADVREQ	Server	A server called the <u>DdePostAdvise</u> function, indicating that the value of a data item in an advise loop had changed.
XTYP_ADVSTART	Server	A client specified the XTYP_ADVSTART transaction type in a call to the <u>DdeClientTransaction</u> function.
XTYP_ADVSTOP	Server	A client specified the XTYP_ADVSTOP transaction type in a call to the <u>DdeClientTransaction</u> function.
XTYP_CONNECT	Server	A client called the <u>DdeConnect</u> function, specifying a service name and topic name supported by the server.
XTYP_CONNECT_CONFIRM	Server	The server returned TRUE in response to an XTYP_CONNECT or XTYP_WILDCONNECT transaction.
XTYP_DISCONNECT	Client/Server	A partner in a conversation called the <u>DdeDisconnect</u> function, causing both partners to receive this transaction.
XTYP_ERROR	Client/Server	A critical error has occurred. The DDEML may not have sufficient resources to continue.
XTYP_EXECUTE	Server	A client specified the XTYP_EXECUTE transaction type in a call to the <u>DdeClientTransaction</u> function.
XTYP_MONITOR	DDE monitoring application	A DDE event occurred in the system.
XTYP_POKE	Server	A client specified the XTYP_POKE transaction type in a call to the <u>DdeClientTransaction</u> function.
XTYP_REGISTER	Client/Server	A server application used the <u>DdeNameService</u> function to register a service name.
XTYP_REQUEST	Server	A client specified the XTYP_REQUEST transaction type in a call to the <u>DdeClientTransaction</u> function.

XTYP_UNREGISTER	Client/Server	A server application used the <u>DdeNameService</u> function to <u>unregister a service name</u> .
XTYP_WILDCONNECT	Server	A client called the <u>DdeConnect</u> or <u>DdeConnectList</u> function, specifying NULL for the service name, the topic name, or both.
XTYP_XACT_COMPLETE	Client	An asynchronous transaction, sent when the client specified the TIMEOUT_ASYNC flag in a call to the <u>DdeClientTransaction</u> function, <u>has concluded</u> .



## DDEML Error Detection

Whenever a DDEML function fails, an application can call the DdeGetLastError function to determine the cause of the failure. The DdeGetLastError function returns an error value that specifies the cause of the most recent error.

## Monitoring Applications

Microsoft Windows DDESpy (DDESPY.EXE) monitors DDE activity in the system. You can use DDESpy as a tool for debugging your DDE applications.

You can use the API elements of the DDEML to create your own DDE monitoring applications. Like any DDEML application, a DDE monitoring application contains a DDE callback function. The DDEML notifies a monitoring application's DDE callback function whenever a DDE event occurs, passing information about the event to the callback function. The application typically displays the information in a window or writes it to a file.

To receive notifications from the DDEML, an application must have registered itself as a DDE monitor by specifying the APPCLASS\_MONITOR flag in a call to the DdeInitialize function. In this same call, the application can specify one or more monitor flags to indicate the types of events of which the DDEML is to notify the application's callback function. The following table describes each of the monitor flags an application can specify:

Flag	Meaning
MF_CALLBACKS	Notifies the callback function whenever a transaction is sent to any DDE callback function in the system.
MF_CONV	Notifies the callback function whenever a conversation is established or terminated.
MF_ERRORS	Notifies the callback function whenever a <u>DDEML</u> error occurs.
MF_HSZ_INFO	Notifies the callback function whenever a <u>DDEML</u> application creates, frees, or increments the use count of a string handle or whenever a string handle is freed as a result of a call to the <u>DdeUninitialize</u> function.
MF_LINKS	Notifies the callback function whenever an advise loop is started or ended.
MF_POSTMSGS	Notifies the callback function whenever the system or an application posts a DDE message.
MF_SENDMSGS	Notifies the callback function whenever the system or an application sends a DDE message.

The following example shows how to register a DDE monitoring application so that its DDE callback function receives notifications of all DDE events:

```
DWORD idInst;  
PFNCALLBACK lpDdeProc;  
hInst = hInstance;  
  
lpDdeProc = (PFNCALLBACK) MakeProcInstance(  
    (FARPROC) DDECallback, /* points to callback function */  
    hInstance); /* instance handle */  
  
if (DdeInitialize(  
    (LPDWORD) &idInst, /* instance identifier */  
    lpDdeProc, /* points to callback function */  
    APPCLASS_MONITOR | /* this is a monitoring application */  
    MF_CALLBACKS | /* monitor callback functions */  
    MF_CONV | /* monitor conversation data */  
    MF_ERRORS | /* monitor DDEML errors */  
    MF_HSZ_INFO | /* monitor data-handle activity*/  
    MF_LINKS | /* monitor advise loops */  
    MF_POSTMSGS | /* monitor posted DDE messages */  
    MF_SENDMSGS, /* monitor sent DDE messages */  
    0L)) /* reserved*/  
    return FALSE;
```

The DDEML informs a monitoring application of a DDE event by sending an XTYP\_MONITOR transaction to the application's DDE callback function. During this transaction, the DDEML passes a monitor flag that specifies the type of DDE event that has occurred and a handle of a global memory object that contains detailed information about the event. The DDEML provides a set of structures that the application can use to extract the information from the memory object. There is a corresponding structure for each type of DDE event. The following table describes each of these structures:

Structure	Description
-----------	-------------

<u>MONCBSTRUCT</u>	Contains information about a transaction.
<u>MONCONVSTRUCT</u>	Contains information about a conversation.
<u>MONERRSTRUCT</u>	Contains information about the latest DDE error.
<u>MONLINKSTRUCT</u>	Contains information about an advise loop.
<u>MONHSZSTRUCT</u>	Contains information about a string handle.
<u>MONMSGSTRUCT</u>	Contains information about a DDE message that was sent or posted.

The following example shows the DDE callback function of a DDE monitoring application that formats information about each string handle event and then displays the information in a window. The function uses the MONHSZSTRUCT structure to extract the information from the global memory object.

```

HDDEDATA CALLBACK DDECallback(wType, wFmt, hConv, hsz1, hsz2,
    hData, dwData1, dwData2)
WORD wType;
WORD wFmt;
HCONV hConv;
HSZ hsz1;
HSZ hsz2;
HDDEDATA hData;
DWORD dwData1;
DWORD dwData2;
{
    LPVOID lpData;
    char *szAction;
    char buf[256];
    DWORD cb;

    switch (wType) {
    case XTYP_MONITOR:

        /* Obtain a pointer of the global memory object. */
        if (lpData = DdeAccessData(hData, &cb)) {

            /* Examine the monitor flag. */

            switch (dwData2) {
            case MF_HSZ_INFO:

#define PHSZS ((MONHSZSTRUCT FAR *)lpData)

                /*
                * The global memory object contains
                * string-handle data. Use the MONHSZSTRUCT
                * structure to access the data.
                */

                switch (PHSZS->fsAction) {

                    /*
                    * Examine the action flags to determine
                    * the action performed on the handle.
                    */

                    case MH_CREATE:
                        szAction = "Created";
                        break;

                    case MH_KEEP:
                        szAction = "Incremented";
                        break;

                    case MH_DELETE:

```

```

szAction = "Deleted";
break;

    case MH_CLEANUP:
szAction = "Cleaned up";
break;

    default:
DdeUnaccessData(hData);
return ((HDDEDATA) 0);
}

    /* Write formatted output to a buffer. */

    wsprintf(buf,
"Handle %s, Task: %x, Hsz: %lx(%s)",
(LPSTR) szAction, PHSZS->hTask, PHSZS->hsz,
(LPSTR) PHSZS->str);
.
. /* Display text in window or write to file. */
.

    break;

#undef PHSZS

.
. /* Process other MF_* flags. */
.

default:
    break;
}
}

    /* Free the global memory object. */

DdeUnaccessData(hData);
break;

    default:
break;
}
return ((HDDEDATA) 0);
}
}

```

## Object Linking and Embedding Overview (3.1)

This topic describes the implementation of object linking and embedding (OLE) for applications that run with the Microsoft Windows operating system. The topic also describes how an application can use linked and embedded objects to create compound documents. The following topics discuss the behavior and implementation of object linking and embedding:

[Compound Documents](#)

[Data Transfer in OLE](#)

[Client Applications](#)

[Server Applications](#)

[Object Handlers](#)

[Direct Use of Dynamic Data Exchange](#)

This topic does not go into detail about the recommended user interface for applications that use linked and embedded objects.

# Compound Documents

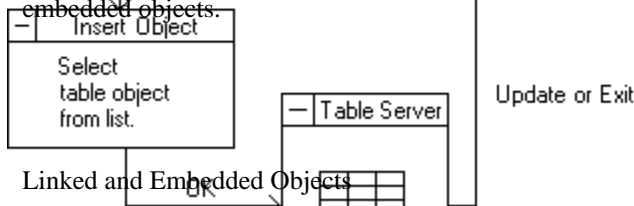
## Compound Documents

An application that uses OLE can cooperate with other OLE applications to produce a document containing different kinds of data, all of which can be easily manipulated by the user. The user editing such a document is able to improve the document by using the best features of many different applications. An application that implements OLE gives its users the ability to move away from an application-centered view of computing and toward a document-centered view. In application-centered computing, the tool used to complete a task is often a single application; whereas, in document-centered computing, a user can combine the advantages of many tools to complete a job.

A document that uses linked and embedded objects can contain many kinds of data in many different formats; such a document is called a compound document. A compound document uses the facilities of different OLE applications to manipulate the different kinds of data it displays. Any kind of data format can be incorporated into a compound document, with little or no extra code. OLE applications can even support data formats that have not yet been invented. The user working with a compound document does not need to know which data formats are compatible with one another or how to find and start any applications that created the data. Whenever a user chooses to work with part of a compound document, the application responsible for that part of the document starts automatically.

For example, a compound document could be a brochure that included text, charts, ranges of cells in a spreadsheet, and illustrations. The information could be embedded in the document, or the document could contain links to certain information instead of containing the information itself. The user working with the brochure could automatically switch between the applications that produced its components.

The following illustration shows the relationships between a compound document and its linked and embedded objects.



An object is any data that can be presented in a compound document and manipulated by a user. Anything from a single cell in a spreadsheet to an entire document can be an object. When an object is incorporated into a document, it maintains an association with the application that produced it. That association can be a link, or the object can be embedded in the file.

If the object is linked, the document provides only minimal storage for the data to which the object is linked, and the object can be updated automatically whenever the data in the original application changes. For example, if a range of spreadsheet cells were linked to information in a text file, the data would be stored in some other file and only a link to the data would be saved with the text file.

If an object is embedded, all the data associated with it is saved as part of the file in which it is embedded. If a range of spreadsheet cells were embedded in a text file, the data in the cells would be saved with the text file, including any necessary formulas; the name of the server for the spreadsheet cells would be saved along with this data. The user could select this embedded object while working with the text file, and the spreadsheet application would be started automatically for editing those cells. The presentation and the behavior of the data is the same for a linked and an embedded object.

## Packages

A package is a type of OLE object that encapsulates another object, a file, or a command line inside a graphic representation (such as an icon or bitmap). When the user double-clicks the graphic object, the OLE libraries activate the object inside the package. The package itself is always an embedded object, not a link. The contents of a package can be an embedded object, a link, or even a file dropped from Windows File Manager.

Packages are useful for presenting compact token views of large files or OLE objects. An application could also use a package as it would use a hyperlink--that is, to connect information in different documents.

Windows version 3.1 includes the application Microsoft Windows Object Packager (PACKAGER.EXE). With Packager, a user can associate a file or data selection with an icon or graphic.

## Verbs

The types of actions a user can perform on an object are called verbs. Two typical verbs for an object are Play and Edit.

The nature of an object determines its behavior when a user works with it. The most typical use for some objects, such as voice annotations and animated scripts, is to play them. For example, a user could play an animated script by double-clicking it. In this case, Play is the primary verb for the object.

For other objects, the most typical use is to edit them. In the case of text produced by a word processor, for example, the primary verb could be Edit.

The client application typically specifies the primary verb when the user double-clicks an object. However, the server application determines the meaning of that verb. A user can invoke an object's subsidiary verbs by using the Class Name Object command or the Links dialog box. For more information about these topics, see Client User Interface.

The action taken when a user double-clicks a package is that of the primary verb of the object inside the package. The secondary verb for a packaged object is Edit Package; when the user chooses this verb, Packager starts. The user can use Packager to gain access to the secondary verb for the object inside the package.

Many objects support only one verb--for example, an object created by a text editor might support only Edit. If an object supports only one verb, that verb is used no matter what the client application specifies.

For more information about verbs, see Registration.

### Benefits of Object Linking and Embedding

OLE offers the following benefits:

- An application can specialize in performing one job very well. For example, a drawing application that implements OLE does not need any text-editing tools; a user could put text into the drawing and edit that text by using any text editor that supports OLE.
- An application is automatically extensible for future data formats, because the content of an object does not matter to the containing document.
- A user can concentrate on the task instead of on any software required to complete the task.
- A file can be more compact, because linking to objects allows a file to use an object without having to store that object's data.
- A document can be printed or transmitted without using the application that originally produced the document.
- Linked objects in a file can be updated dynamically.

Future implementations of this protocol could take advantage of a wide variety of object types. For example, the user of a voice-recorder application could dictate a comment, package the comment as an object with a visual representation, and embed the graphic as an object in a text file. When a user double-clicked the graphic for this object (a pair of lips, perhaps), the voice-recorder application would play the recorded comment. Linked and embedded objects also lend themselves to implementations such as animated drawings, executable macro scripts, hypertext, and annotations.

### Choosing Between OLE and the DDEML

Applications can exchange data by using either OLE or the DDEML. Unless an application has a strong requirement for managing multiple items in a single conversation with another application, the application should use OLE instead of the DDEML.

Both OLE and the DDEML are message-based systems supported by dynamic-link libraries. Developers are encouraged to use these libraries rather than using the underlying message-based protocols. For more information about the message-based OLE protocol, see Direct Use of Dynamic Data Exchange.

Unlike OLE, the DDEML supports multiple items per conversation. With OLE, a client needing links to several objects in a document must establish a separate conversation for each object.

OLE offers the following advantages that the DDEML does not:

Advantage	Description
Extensibility to future enhancements	The OLE libraries may be updated in future releases to support new data formats, link tracking, editing without exiting the client application, and other enhancements that will not be immediately available to applications that use the DDEML.
Persistent embedding and linking of objects	The OLE libraries do most of the work of activating objects when an embedded document is reopened,

by reestablishing the conversation between a client and server. In contrast, establishing a DDE link (DDE advise loop) is the responsibility of either the user (if the link is not persistent) or of the application (if the link is persistent).

Rendering of common data formats

The OLE libraries assume the burden of rendering common data formats on a display context. DDE applications, however, must do this work themselves.

Server rendering of specialized data formats

The OLE libraries facilitate the rendering of specialized data formats in the client's display context. (The server application or object handler actually performs the rendering.) The client application has to do very little work to render the embedded or linked data in its display context. Such rendering of embedded or linked data is beyond the scope of the DDEML alone.

Activating embedded and linked objects

The OLE libraries support activating a server to edit a linked or embedded object or to render data. Activating servers for data rendering and editing is beyond the scope of the DDEML.

Creating objects and links from the clipboard

The OLE libraries do most of the work when an application is using the clipboard to copy and paste links or exchange objects. In contrast, DDE applications must call the Windows clipboard functions directly to perform such operations.

Creating objects and links from files

The OLE libraries provide direct support for using files to exchange data. No DDE protocol is defined for this purpose.

The OLE libraries use DDE messages instead of the DDEML, because the libraries were written before the DDEML was available.

#### Using OLE for Standard DDE Operations

Although most of the OLE application programming interface (API) was designed for linked and embedded objects, it can also be applied to standard DDE items. In particular, an application can use the OLE API to perform the following DDE tasks:

- Initializing conversations based on application and topic names or wildcards.
- Requesting data for named items in negotiated formats from a server.
- Establishing an advise loop--that is, requesting that a DDE server notify the client of changes to the values of specified items and, optionally, that the server send the data when the change occurs.
- Sending data from a server to a client.
- Poking data from a client to a server.
- Sending a DDE command. (This is supported by the OleExecute function.)

An OLE client application receives a pointer to an OLEOBJECT structure; this structure includes class name, document name, and item name information. These names correspond exactly to DDE counterparts, as follows:

OLE name	DDE name
Class name	Service name (formerly called "application name")
Document name	Topic name
Item name	Item name

The client can use the OleCreateFromFile function to make an object and specify all three names. If the client application needs multiple items from the same topic, it must have an OLEOBJECT structure for each item, which causes a DDE conversation to be created for each item.

The client library maps OLE functions that work on the OLEOBJECT structure to DDE messages as follows:

OLE function	DDE message
<u>OleExecute</u>	<u>WM_DDE_EXECUTE</u>
<u>OleRequestData</u>	<u>WM_DDE_REQUEST</u>
<u>OleSetData</u>	<u>WM_DDE_POKE</u>



Some functions (such as OleActivate) are too complicated for this one-to-one mapping of function to DDE message. For these functions, the DDE message depends on the circumstance.

If a client application needs to duplicate the functionality of WM\_DDE\_ADVISE with OLE, the client must create the link with olerender\_format for the renderopt parameter, specify the required format, and use the OleGetData function to retrieve the value when the callback function receives the OLE\_CHANGED notification. If more than one item or format is required, the client must create an OLEOBJECT structure for each item/format pair. Although this method creates a conversation for each advise transaction, it may be inefficient if the client needs to create many such conversations.

A server application can make itself accessible to DDE by calling the OleRegisterServer function to make the System topic available and the OleRegisterServerDoc function to make other topics available. When a client connects and asks for an item, the server library calls the GetObject function in the server's OLESERVERDOCVTBL structure, followed by other server-implemented functions that are appropriate to the client's request. (Usually, the library calls the GetData function in the server's OLEOBJECTVTBL structure.) As long as the object allocated by the call to GetObject has not been released, the server should send a notification when the item has changed, so that the OLE libraries can send data to clients that have sent WM\_DDE\_ADVISE.

#### Using Both OLE and the DDEML

Some applications may need features supported only by OLE and may also need to use the DDEML to support simultaneous links for many items that are updated frequently. Client applications of this kind can use the OLE libraries to initiate conversations with OLE servers and the DDEML to initiate conversations with DDE servers.

Server applications that need to support both OLE and the DDEML must use different service names (DDE application names) for OLE and DDE conversations; otherwise, the OLE and DDEML libraries cannot determine which library should respond when an initiation request is received. Typically, the application changes the service name for the OLE conversation in this case, because other applications and the user must use the service name for the DDE conversation, but the OLE service name is hidden.

## Data Transfer

### Data Transfer in OLE

This section gives a brief overview of how applications share information under OLE. Details of the implementation are given in later sections of this topic.

Applications use three dynamic-link libraries (DLLs), OLECLI.DLL, OLESVR.DLL, and SHELL.DLL, to implement object linking and embedding. Object linking and embedding is supported by OLECLI.DLL and OLESVR.DLL. The registration database is supported by SHELL.DLL.

### Client Applications

An OLE client application can accept, display, and store OLE objects. The objects themselves can contain any kind of data. A client application typically identifies an object by using a distinctive border or other visual cue, as described in Microsoft Windows User Interface Guidelines.

### Server Applications

An OLE server is any application that can edit an object when the OLE libraries inform it that the user of a client application has selected the object. (Some servers can perform operations on an object other than editing.) When the user double-clicks an object in a client application, the server associated with that object starts and the user works with the object inside the server application. When the server starts, its window is typically sized so that only the object is visible. If the user double-clicks a linked object, the entire linked file is loaded and the linked portion of the file is selected. For embedded objects, the user chooses the Update command from the File menu to save changes to the object and chooses Exit when finished.

Many applications are capable of acting as both clients and servers for linked and embedded objects.

### Object Handlers

Some OLE server applications implement an additional kind of OLE library called an object handler. Object handlers are dynamic-link libraries that act as intermediaries between client and server applications. Typically, an object handler is supplied by the developers of a server application as a way of improving performance. For example, an object handler could be used to redraw a changed object if the presentation data for that object could not be rendered by the client library.

### Communication Between OLE Libraries

Client applications use functions from the OLE API to inform the client library, OLECLI.DLL, that a user wants to perform an operation on an object. The client library uses DDE messages to communicate with the server library, OLESVR.DLL. The server library is responsible for starting and stopping the server application, directing the interaction with the server's callback functions, and maintaining communication with the client library.

When a server application modifies an embedded object, the server notifies the server library of changes. The server library then notifies the client library, and the client library calls back to the client application, informing it that the changes have occurred. Typically, the client application then forces a repaint of the embedded object in the document file. If the server changes a linked object, the server library notifies the client library that the object has changed and should be redrawn.

### Clipboard Conventions

When first embedding or linking an object, OLE client and server applications typically exchange data by using the clipboard. When a server application puts an object on the clipboard, it represents the object with data formats, such as Native data, OwnerLink data, ObjectLink data, and a presentation format. The order in which these formats are put on the clipboard is very important, because the order determines the type of object. For example, if the first format is Native and the second is OwnerLink, client applications can use the data to create an embedded object. If the first format is OwnerLink, however, the data describes a linked object.

Native data completely defines an object for a particular server. The data can be meaningful only to the server application. The client application provides storage for Native data, in the case of embedded objects.

OwnerLink data identifies the owner of a linked or embedded object.

Presentation formats allow the client library to display the object in a document. CF\_METAFILEPICT, CF\_DIB, and CF\_BITMAP are typical presentation formats. Native data can be used as a presentation format, typically when an object handler has been defined for that class of data. Native data cannot be

used twice in the definition of an object, however; if the server puts Native and OwnerLink data on the clipboard to describe an embedded object, it cannot use Native data as a presentation format for that object. The ability of object handlers to use Native data as the presentation data accounts for the significance of the order of the formats: the order is the only way to distinguish between an embedded object and a link that uses Native data for its presentation.

ObjectLink data identifies a linked object's class and document and the item that is the source for the linked object. (If the item name specified in the ObjectLink format is NULL, the link refers to the entire server document.)

The following table describes the contents of the ObjectLink, OwnerLink, and Native clipboard formats:

Format name	Contents
ObjectLink	Null-terminated string for class name, null-terminated string for document name, string for item name with two terminating null characters.
OwnerLink	Null-terminated string for class name, null-terminated string for document name, string for item name with two terminating null characters.
Native	Stream of bytes interpreted only by the server application or object-handler library. This format can be unique to the server application and must allow the server to load and work with the object.

Although the ObjectLink and OwnerLink formats contain the same information, the OLE libraries use them differently. The libraries use OwnerLink format to identify the owner of an object (which can be different from the source of the object) and ObjectLink format to identify the source of the data for an object.

The class name in the ObjectLink or OwnerLink format is a unique name for a class of objects that a server supports. Server applications register the class name or names they support in the registration database. (For example, the class name used by Windows Paintbrush™ is PBrush.) An application can use the class name to look up information about a server in the registration database. (For more information about registration, see Registration.) The document name is typically a fully qualified path that identifies the file containing a document. The item name uniquely identifies the part of a document that is defined as an object. Item names are assigned by server applications; an item name can be any string that the server uses to identify part of a document. Items names cannot contain the forward-slash (/) character.

Data in OwnerLink or ObjectLink format could look like the following example:

```
Microsoft Excel Worksheet\0c:\directry\docname.xls\0R1C1:R5C3\0\0
```

The order in which various data formats are put on the clipboard depends on the type of data being copied to the clipboard and the capabilities of the server application. The following table shows the order of clipboard data formats for four different types of data selections. An object does not necessarily use all of the formats listed for it.

Source selection	Clipboard contents, in order
Embedded object	Native& OwnerLink& Picture or other presentation format (optional)& ObjectLink (included only if the server also supports links)
Linked object	OwnerLink& Picture or other presentation format (optional; for linked objects, this can be Native data)& ObjectLink
Pictorial data	Application-specific formats& Native& OwnerLink& Picture& ObjectLink
Structured data	Structured data formats (if selection is structured data only)& Native& OwnerLink& Picture, text, and so on& ObjectLink

Before copying data for an embedded or linked object to the clipboard, a server puts descriptions of the data formats on the clipboard. These data formats are listed in order of their level of description, from most descriptive to least. (For example, Microsoft Word would put rich-text format (RTF) onto the clipboard first, then the CF\_TEXT clipboard format.)

When a user chooses the Paste command, the client application queries the formats on the clipboard and uses the first format that is compatible with the destination for the object. Because server applications put data onto the clipboard in order of their fidelity of description, the first acceptable format found by a client application is the best format for it to use. If the client application finds an acceptable format prior to the Native format, it incorporates the data into the target document without making it an embedded object. (For example, a Microsoft Word document would not make an embedded object from clipboard data that was in RTF format. Similarly, structured data or a structured document would be embedded into a drawing application but would be converted into the destination document's native data type if the destination were a worksheet or structured document.) If the client application cannot accept any of the data formats prior to Native and OwnerLink, it uses the Native and OwnerLink formats to make an embedded object and

then finds an appropriate presentation format. The destination application may require different formats depending on where the selection is to be placed in the destination document; for example, pasting into a picture frame and pasting into a stream of text could require different formats.

When a user chooses the Paste Link command from the Edit menu, the client application looks for the ObjectLink format on the clipboard and ignores the Native and OwnerLink formats. The ObjectLink format identifies the source class, document, and object. If the application finds the ObjectLink format and a useful presentation format, it uses them to make an OLE link to the source document for the object. If the ObjectLink format is not available, the client application may look for the Link format and create a DDE link. This type of link does not support the OLE protocol.

When an application that does not support OLE copies from an OLE item on the clipboard, it ignores the Native, OwnerLink and ObjectLink formats; the behavior of the copying application does not change.

## Registration

The registration database supports linked and embedded objects by providing a systemwide source of information about whether server applications support the OLE protocol, the names of the executable files for these applications, the verbs for classes of objects, and whether an object-handler library exists for a given class of object. For more information about this database, see The Windows Shell Overview.

When a server application is installed, it registers itself as an OLE server with the registration database. (This database is supported by the dynamic-link library SHELL.DLL.) To register itself as an OLE server, a server application records in the database that it supports one or more OLE protocols. The only protocols supported by version 1.x of the Microsoft OLE libraries are StdFileEditing and StdExecute. StdFileEditing is the current protocol for linked and embedded objects. StdExecute is used only by applications that support the OleExecute function. (A third name, Static, describes a picture than cannot be edited by using standard OLE techniques.)

When a client activates a linked or embedded object, the client library finds the command line for the server in the database, appends the /Embedding or /Embedding filename command-line option, and uses the new command line to start the server. Starting the server with either of these options differs from the user starting it directly. Either a slash (/) or a hyphen (-) can precede the word Embedding. For details about how a server reacts when it is started with these options, see Opening and Closing Objects.

The entries in the registration database are used whenever an application or library needs information about an OLE server. For example, client applications that support the Insert Object command refer to the database in order to list the OLE server applications that could provide a new object. The client application also uses the registration database to retrieve the name of the server application for the Paste Special dialog box.

## Registration Database

Applications typically add key and value pairs to the registration database by using Microsoft Windows Registration Editor (REGEDIT.EXE). Applications could also use the registration functions to add this information to the database.

The registration database stores keys and values as null-terminated strings. Keys are hierarchically structured, with the names of the components of the keys separated by backslash characters (\). The class name and server path should be registered for every class the server supports. (This class name must be the same string as the server uses when it calls the `OleRegisterServer` function.) If a class has an object-handler library, it should be registered using the `handler` keyword. An application should also register all the verbs its class or classes support. (An application's verbs must be sequential; for example, if an object supports three verbs, the primary verb is 0 and the other verbs must be 1 and 2.)

To be available for OLE transactions, a server should register the key and value pairs shown in the following example when it is installed. This example shows the form of key and value pairs as they would be added to a database with Registration Editor. Although the text string sometimes wraps to the next line in this example, the lines should not include newline characters when they are added to the database.

```
HKEY_CLASSES_ROOT\class name = readable version of class name
HKEY_CLASSES_ROOT\class name\ext = class name
HKEY_CLASSES_ROOT\class name\protocol\StdFileEditing\server =
    executable file name
HKEY_CLASSES_ROOT\class name\protocol\StdFileEditing\handler =
    dll name
HKEY_CLASSES_ROOT\class name\protocol\StdFileEditing\verb\0 =
    primary verb
HKEY_CLASSES_ROOT\class name\protocol\StdFileEditing\verb\1 =
```

secondary verb

Servers that support the OleExecute function also add the following line to the database:

```
HKEY_CLASSES_ROOT\class name\protocol\StdExecute\server =  
executable file name
```

An ampersand (&) can be used in the verb specification to indicate that the following character is an accelerator key. For example, if a verb is specified as &Edit, the E key is an accelerator key.

A server can register the entire path for its executable file, rather than registering only the filename and arguments. Registering only the filename fails if the application is installed in a directory that is not mentioned in the PATH environment variable. Usually, registering the path and filename is less ambiguous than registering only the filename.

Servers can register data formats that they accept on calls to the OleSetData function or that they can return when a client calls the OleRequestData function. Clients can use this information to initialize newly created objects (for example, from data selected in the client) or when using the server as an engine (for example, when sending data to a chart and getting a new picture back). Client applications should not depend on the requested data format, because the calls can be rejected by the server.

In the following example, format is the string name of the format as passed to the RegisterClipboardFormat function or is one of the system-defined clipboard formats (for example, CF\_METAFILEPICT):

```
HKEY_CLASSES_ROOT\class name\protocol\StdFileEditing  
  \SetDataFormats = format[,format]  
HKEY_CLASSES_ROOT\class name\protocol\StdFileEditing  
  \RequestDataFormats = format[,format]
```

For compatibility with earlier applications, the system registration service also reads and writes registration information in the [embedding] section of the WIN.INI initialization file.

In the following example, the keyword picture indicates that the server can produce metafiles for use when rendering objects:

```
[embedding]  
classname=comment,textual class name,path/arguments,picture
```

#### Version Control for Servers

Server applications should store version numbers in their Native data formats. New versions of servers that are intended to replace old versions should be capable of dealing with data in Native format that was created by older versions. It is sometimes important to give the user the option of saving the data in the old format, to support an environment with a mixture of new and old versions, or to permit data to be read by other applications that can interpret only the old format.

There can be only one application at a time (on one workstation) registered as a server for a given class name. The class name (which is stored with the Native data for objects) and the server application are associated in the registration database when the server application registers during installation.

If a new version of a server application allows the user to keep the old version available, a new class name should be allocated for the new server. A good way to do this is to append a version number to the class name. This allows the user to easily differentiate between the two versions when necessary. (The OLE libraries do not check these numbers.)

When the new version of the server is installed, the user should be given the option of either mapping the old objects to the new server (registering the new server as the server for both class names) or keeping them separate. When the user keeps them separate, the user will be aware of two kinds of object (for example, Graph1 and Graph2).

The user should be able to discard the old server version at a later time by remapping the registration database, typically with the help of the server setup program. To remap the database, the old and new objects are given the same value for readable version of class name (although their class names remain distinct). The OLE client library removes duplicate names when it produces the list in the Insert Object dialog box. When a client application produces a list by enumerating the registration database, the application must do this filtering itself.

#### Client User Interface

When a user opens a document that contains a linked or embedded object, the client application uses the OLE functions to communicate with OLECLI.DLL. This library assists the client application with such

tasks as loading and drawing objects, updating objects (when necessary), and interacting with server applications.

#### New and Changed Commands

An OLE client application typically implements the following new or changed commands as part of its Edit menu. (Although this user interface is not mandatory, it is recommended for consistency with existing OLE applications.)

Command	Description
Copy	Copies an object from a document to the clipboard.
Cut	Removes an object from a document and places it on the clipboard.
Paste	Copies an object from the clipboard to a document.
Paste Link	Inserts a link between a document and the file that contains an object.
Class Name Object	Makes it possible for the user to activate the verbs for a linked or embedded object. The actual text used instead of the Class Name placeholder depends upon the selected object.
Links	Makes it possible for the user to change link updating options, update linked objects, cancel links, repair broken links, and activate the verbs associated with linked objects.
Insert Object	Starts the server application chosen by the user from a dialog box and embeds in a document the object produced by the server. This command is optional.
Paste Special	Transfers an object from the clipboard to a document or inserts a link to the object, using the data format chosen by the user from a dialog box. This command is optional.

In addition to the listed menu changes, client applications must also implement changes to their Copy and Cut commands. When a linked or embedded object is selected in the client application, the application can use the OleCopyToClipboard function to implement the Cut and Copy commands.

When the user chooses the Paste command, a client application should insert the contents of the clipboard at the current position in a document. If the clipboard contains an object, choosing this command typically embeds the object in the document.

When the user chooses the Paste Link command, the client library typically inserts a linked object at the current position in a document. The object is displayed in the document, but the Native data that defines that object is stored elsewhere.

If a user copies a linked object to the clipboard, other documents can use this object to produce a link to the original data.

The Class Name Object command allows the user to choose one of an object's verbs. If the selection in the document is an embedded object, the Class Name placeholder is typically replaced by the class and name of the object; for example, if a user selects an object that is a range of spreadsheet cells for Microsoft Excel, the text of the command might be "Microsoft Excel Worksheet Object." If an object supports only one verb, the name of the verb should precede the class name in the menu item; for example, if the only verb for a text object is Edit, the text of the command might be "Edit WPDocument Object." When an object supports more than one verb, choosing the Class Name Object command brings up a cascading menu listing each of the verbs.

For more information about verbs, see Verbs.

Choosing the Links command brings up a Links dialog box, which lists the selected links and their source documents and gives the user the opportunity to change how the links are updated, cancel the link, change the link, or activate the verbs for the link. A user can use this dialog box to repair links to objects that have been moved or renamed.

When the user chooses the Paste Special command, a client application should bring up a dialog box listing the data formats the client supports that are presently on the clipboard. The Paste Special dialog box makes it possible for the user to override the default behaviors of the Paste and Paste Link commands. For example, if the first format on the clipboard can be edited by the client application, the default behavior is for the client to copy the data into the document without making it into an object. The user could override this default behavior and create an object from such data by using the Paste Special command.

When the user chooses the Insert Object command, a client application should allow the user to insert an object of a specified class at the current position in a document. For example, to insert a range of spreadsheet cells in a text document, a user could choose the Insert Object command and select "Microsoft Excel Worksheet" from the dialog box. Selecting this item would start Microsoft Excel. The user would

use Microsoft Excel to create the object to be embedded in the text document. When finished, the user would quit Microsoft Excel; the range of spreadsheet cells would automatically be embedded in the text document.

The Insert Object command is optional because a user could achieve the same results without it, although the procedure is less convenient. To use the same example as that shown in the preceding paragraph, the user could leave the client application, start Microsoft Excel, and use the Microsoft Excel Cut or Copy command to transfer data to the clipboard. After returning to the client application, the user could choose the Paste command to move the data from the clipboard into the text document.

If the user chooses the Undo command after activating an object, all of the changes made since the object was last updated (or since the object was activated, if it has not been updated) are discarded and the object returns to its state prior to the update. The Undo command closes the connection to the server.

For more information about these commands, including illustrations of the dialog boxes, see Microsoft Windows User Interface Guidelines.

### Using Packages

A package is an embedded graphical object that contains another object, which can be linked or embedded. For example, a user can package a file in an icon and embed the icon in an OLE document. Most of the packaging capabilities are provided by the dynamic-link library SHELL.DLL.

A user can put a package into an OLE document in a number of different ways:

- Copy a file from File Manager to the clipboard, and then choose the Paste or Paste Link command from the Edit menu in the client application.
- Drag a file from File Manager and drop it in the open window for a document in a client application.
- Select Package from the list of objects in the Insert Object dialog box. This starts Object Packager, with which the user can associate a file or data selection with an icon or graphic. Choosing Update and then Exit from Object Packager's File menu puts the package in the client document.
- Run Packager directly, following the steps outlined in the previous list item.

For information about how a client application should react when a user drops a file from File Manager in the client's window, see the description of the [OleCreateFromFile](#) function.

A user whose system does not include the Windows version 3.1 File Manager can follow these steps to create a package by using Object Packager:

- Copy to the clipboard the data to be packaged.
- Open Object Packager and paste the data into it. (At this point, the user could modify the default icon, the default label identifying the icon, or both.)
- Choose Copy Package from the Object Packager Edit menu to copy the package to the clipboard.
- Choose the Paste command from the Edit menu in the client application to embed the package.

For more information about Object Packager, see Packages, or Microsoft Windows User Interface Guidelines.

### Server User Interface

A server for linked and embedded objects is any application that can be used to edit an object when the OLE libraries inform it that the user of a client application has activated the object. (Some servers can use verbs other than Edit to work with an object.) Although client applications implement many changes to the user interface to support OLE, the user interface does not change significantly for server applications.

OLE servers typically implement changes to the following commands in the Edit menu. (Although this user interface is not mandatory, it is recommended for consistency with existing OLE applications.)

Command	Description
Cut	Transfers data from the application to the clipboard, deleting the data from the source document. A client application can use this data to create an embedded object.
Copy	Transfers a copy of the data from the application to the clipboard. A client application can use this data to create an embedded object and may be able to establish a link to the source document.

Some menu items change names or behave differently when a server is started as part of activating an object from within a compound document. The exact behavior of the server depends on whether the server supports the multiple document interface (MDI).

### Updating Objects from Multiple-Instance Servers

When an embedded object is edited or played by a multiple-instance server--that is, a server that does not support the multiple document interface (MDI), the Save command on the File menu should change to Update. (This change does not occur when a server starts for a linked object.) When the user chooses the Update command, the object in the client is updated but the focus remains with the server window. To close the server window, the user chooses the Exit command.

When the user chooses the Save As, New, or Open command, the application should display a warning message asking the user whether to update the object in the compound document before performing the action. The New and Open commands break the link between the client and server applications. The Save As command also breaks the link between the client and server if the server was editing an embedded object.

#### Updating Objects from Single-Instance Servers

The same rules for updating objects from multiple-instance servers apply to single-instance (MDI) servers, with the following differences:

- When the focus in an MDI server changes from a window in which an embedded object was activated to a window in which a document that does not contain an embedded object is being edited, the Update command should change back to Save.
- When the user chooses the New or Open command, the window containing the embedded object remains open. (This eliminates the need to prompt the user to update the object.)

#### Object Storage Formats

The presentation data in linked or embedded objects can be thought of as a presentation object. A presentation objects can be standard, generic, or NULL. A standard presentation object is used when the format is metafile, bitmap, or device-independent bitmap (DIB). The client library supports the presentation objects, including drawing them. Neither client applications nor object handlers can use the presentation objects; they are solely for the use of the client library.

The following list gives the storage format for strings in OLE. The items appear in the order listed.

Type	Description
LONG	Length of string, including terminating null character.
Variable	Null-terminated stream of bytes.

The following list gives the storage format for the standard presentation object used for linked and embedded objects. The items appear in the order listed.

Type	Description
LONG	OLE version number.
LONG	Format identifier. This value is 5.
Variable	Class string. For standard presentation objects, this string is <u>METAFILEPICT</u> , <u>BITMAP</u> , or <u>DIB</u> .
LONG	Width of object, in MM_HIMETRIC units.
LONG	Height of object, in MM_HIMETRIC units.
LONG	Size of presentation data, in bytes.
Variable	Presentation data.

The following list gives the storage format for the generic presentation object used for linked and embedded objects. Generic objects are used when the clipboard format is other than metafile, bitmap, or DIB. The items appear in the order listed.

Type	Description
LONG	OLE version number.
LONG	Format identifier. This value is 5.
Variable	Class string.
LONG	Clipboard format value. If this value exists, the next item in storage is the size of the presentation data.
LONG	Clipboard format name. This value exists only if the clipboard format value is NULL.
LONG	Size of presentation data, in bytes.
Variable	Presentation data.

The following list gives the storage format for embedded objects. The items appear in the order listed.

Type	Description
LONG	OLE version number.



LONG	Format identifier. This value is 2.
Variable	Class string.
Variable	Topic string.
Variable	Item string.
LONG	Size of Native data, in bytes.
Variable	Native data.
Variable	Presentation object (standard, generic, or NULL).

The following list gives the storage format for linked objects. The items appear in the order listed.

Type	Description
LONG	OLE version number.
LONG	Format identifier. This value is 1.
Variable	Class string.
Variable	Topic string.
Variable	Item string.
Variable	Network name string.
short	Network type.
short	Network driver version number.
LONG	Link update options.
Variable	Presentation object (standard, generic, or NULL).

The following list gives the storage format for static objects. The only difference between the format for static objects and the format for standard presentation objects is the value of the format identifier. The items appear in the order listed.

Type	Description
LONG	OLE version number.
LONG	Format identifier. This value is 3.
Variable	Class string. For static objects, this string is <u>METAFILEPICT</u> , <u>BITMAP</u> , or DIB.
LONG	Width of object, in MM_HIMETRIC units.
LONG	Height of object, in MM_HIMETRIC units.
LONG	Size of presentation data, in bytes.
Variable	Presentation data.

## Client Applications

### Client Applications

A client application uses a server application to activate and render an object contained by a compound document. A client application provides storage for embedded objects, such contextual information as the target printer and page position, and a means for the user to activate the object and the server application associated with that object. Client applications also provide ways of putting embedded and linked objects into a document and taking them out again.

Client applications must provide permanent storage for objects in the compound document's file. When an item being saved is an embedded object, the client library stores the object's Native data, the presentation data for the object (for example, a metafile), and the OwnerLink information. When the item being saved is a link to another document, the client library stores the presentation data and the ObjectLink format.

Client applications accommodate asynchronous operations by defining a callback function to which the library sends notifications about current operations. As long as the client continues to dispatch messages, it can react to the notifications being sent to the callback function and to input from the user. For more information about asynchronous operations, see Asynchronous Operations.

### Starting a Client Application

When a client application starts, it should follow these steps:

- 1 Register the clipboard formats that it requires.
- 2 Allocate and initialize as many OLECLIENT structures as required.
- 3 Allocate and initialize an OLESTREAM structure.

A client application can register the clipboard formats by calling the RegisterClipboardFormat function for each format, specifying such formats as Native, OwnerLink, ObjectLink, and any other formats it requires.

A client application uses two structures to receive information from the client library: OLECLIENT and OLESTREAM.

The OLECLIENT structure points to an OLECLIENTVTBL structure, which in turn points to a callback function supplied by the client application. The OLE libraries use this callback function to notify the client of any changes to an object. The parameters for the callback function are a pointer to the client structure, a pointer to the relevant object, and a value giving the reason for the notification. Typically, an application creates one OLECLIENT structure for each OLEOBJECT structure. Having a separate OLECLIENT structure for each object allows an application to take object-specific action in response to the OLE\_QUERY\_PAINT callback notification.

The OLECLIENT structure can also point to data that describes the state of an object. This data, when present, is supplied and used only by the client application. The client application allocates a separate OLECLIENT structure for each object and stores state information about that object in the structure. Because one argument to the callback function is a pointer to the OLECLIENT structure, this is an efficient method of retrieving the object's state information when the callback function is called.

The OLESTREAM structure points to an OLESTREAMVTBL structure, which is a table of pointers to client-supplied functions for stream input and output. The client libraries use these functions when loading and saving objects. A client can customize functions for particular situations, and a client can make such changes as varying the permanent storage for an object; for example, a client could store an object in a database, instead of in a file with the rest of the document.

The client application should create a pointer to the callback function in the OLECLIENTVTBL structure and pointers to the functions in the OLESTREAMVTBL structure by using the MakeProcInstance function. Callback functions should be exported in the module-definition file.

### Opening a Compound Document

To open a compound document, a client application should take the following steps:

- 1 Register the document with the client library.
- 2 Load the document data from a file.
- 3 For each object in the document, call the OleLoadFromStream function.
- 4 List any objects with manual links so that the user can update them. Automatically update any automatic links.

The OleRegisterClientDoc function registers a document with the client library and returns a handle that is used in object-creation functions and document-management functions. (This registration does not involve the registration database.)

A client application should call the OleLoadFromStream function for each object in the document that will be shown on the screen or otherwise activated. (It is often not necessary to load every object in a document immediately when the document is opened.) Parameters for this function include a pointer to the OLECLIENT structure, which is used to locate the client's callback function (and which is sometimes used by the client to store private state information for the object), and a pointer to the OLESTREAM structure. The library calls the Get function in the OLESTREAMVTBL structure to load the object.

### Document Management

A client application should notify the library when it opens, closes, saves, or renames a document, or causes a document to revert to a previously saved state. A client application can use the following functions to accomplish these tasks:

Function	Description
<u>OleRegisterClientDoc</u>	Registers an opened document with the library.
<u>OleRenameClientDoc</u>	Informs the library that a document has been renamed.
<u>OleRevertClientDoc</u>	Informs the library that a document has reverted to a previously saved state.
<u>OleRevokeClientDoc</u>	Informs the library that a document should be closed or no longer exists.
<u>OleSavedClientDoc</u>	Informs the library that a document has been saved.

A client application should also maintain a persistent name for each object. This name should be unique within the scope of the client document and should be stored with the document. This name is specified when the object is created and should persist when the document is saved and reopened. When a client uses the OleRename function to change the name of an object, the new name must also be unique and must be stored with the document.

### Saving a Document

A client application should follow these steps to save a document:

- 1 Save the data for the document in the document's file.
- 2 For each object in the document, call the OleSaveToStream function.
- 3 When the library confirms that all objects have been saved, call the OleSavedClientDoc function.

A client application can call the OleQuerySize function to determine the size of the buffer required to store an object before calling OleSaveToStream.

### Closing a Document

A client application should follow these steps to close a document:

- 1 For each object in the document, call the OleRelease function.
- 2 Use either the OleRevertClientDoc or the OleSavedClientDoc function to register the current state of the document with the library.
- 3 When the library confirms that all objects have been closed, call the OleRevokeClientDoc function.

### Asynchronous Operations

When a client application calls a function that invokes a server application, actions taken by the client and server can be asynchronous. For example, the actions of updating a document and closing a server are asynchronous. Whenever an asynchronous operation begins, the client library returns OLE\_WAIT\_FOR\_RELEASE. When a client application receives this notification, it must wait for the OLE\_RELEASE notification before it quits. If the client cannot take further action until the asynchronous operation finishes, it should enter a message-dispatch loop and wait for OLE\_RELEASE. Otherwise, it should allow the main message loop to continue dispatching messages so that processing can continue.

An application can run only one asynchronous operation at a time for an object; each asynchronous operation must end with the OLE\_RELEASE notification before the next one begins. The client's callback function must receive OLE\_RELEASE for all pending asynchronous operations before calling the OleRevokeClientDoc function.

Some of the object-creation functions return OLE\_WAIT\_FOR\_RELEASE. The client application can continue to work with the document while waiting for OLE\_RELEASE, but some functions (for example, OleActivate) cannot be called until the asynchronous operation has been completed.

If an application calls a function for an object before receiving OLE\_RELEASE for that object, the function may return OLE\_BUSY. The server also returns OLE\_BUSY when processing a new request would interfere with the processing of a current request from a client application or user. When a function returns OLE\_BUSY, the client application can display a message reporting the busy condition at this point or it can enter a loop to wait for the function to return OLE\_OK. (The OLE\_QUERY\_RETRY notification is also sent to the client's callback function when the server is busy; when the callback function returns FALSE, the transaction with the server is ended.) Note that if the server uses the OleBlockServer function to postpone OLE activities, the OLE\_QUERY\_RETRY notification is not sent to the client.

The following example shows a message-dispatch loop that allows a client application to transact messages while waiting for the OLE\_RELEASE notification:

```
while ((olestat = OleQueryReleaseStatus(lpObject)) == OLE_BUSY) {
    if (GetMessage(&msg, NULL, NULL, NULL)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
}
if (olestat == OLE_ERROR_OBJECT) {
    . /* The lpObject parameter is invalid. */
    .
}
else { /* if olestat == OLE_OK */
    . /* The object is released, or the server has terminated. */
    .
}
```

A server application could end unexpectedly while a client is waiting for OLE\_RELEASE. In this case, the client library recovers properly only if the client uses the OleQueryReleaseStatus function, as shown in the preceding example.

The following table shows which OLE functions can return the OLE\_WAIT\_FOR\_RELEASE or OLE\_BUSY value to a client application:

Function	OLE_BUSY	OLE_WAIT_FOR_RELEASE
<u>OleActivate</u>	Yes	Yes
<u>OleClose</u>	Yes	Yes
<u>OleCopyFromLink</u>	Yes	Yes
<u>OleCreate</u>	No	Yes
<u>OleCreateFromClip</u>	No	Yes
<u>OleCreateFromFile</u>	No	Yes
<u>OleCreateFromTemplate</u>	No	Yes
<u>OleCreateLinkFromClip</u>	No	Yes
<u>OleCreateLinkFromFile</u>	No	Yes
<u>OleDelete</u>	Yes	Yes
<u>OleExecute</u>	Yes	Yes
<u>OleLoadFromStream</u>	No	Yes
<u>OleObjectConvert</u>	Yes	No
<u>OleReconnect</u>	Yes	Yes
<u>OleRelease</u>	Yes	Yes
<u>OleRequestData</u>	Yes	Yes
<u>OleSetBounds</u>	Yes	Yes
<u>OleSetColorScheme</u>	Yes	Yes
<u>OleSetData</u>	Yes	Yes
<u>OleSetHostNames</u>	Yes	Yes
<u>OleSetLinkUpdateOptions</u>	Yes	Yes
<u>OleSetTargetDevice</u>	Yes	Yes
<u>OleUnlockServer</u>	No	Yes

OleUpdate

Yes

Yes

### Displaying and Printing Objects

When an object has been loaded and, if necessary, brought up to date, the object can be displayed or printed with the container document. To display an object, the client application should set up the device context and bounding rectangle (ensuring that they use the same mapping mode) and then call the OleDraw function. The client application can use the OleQueryBounds function to retrieve the size of the bounding rectangle on the target device.

An object handler can be used to draw an object. If an object handler exists for an object, the call to the OleDraw function is received and processed by the object handler. If there is no object handler, the client library uses the object's presentation data to display or print the object.

If the presentation data for an object is a metafile, the library periodically sends an OLE\_QUERY\_PAINT notification to the client's callback function while drawing the object. If the callback function returns FALSE, the OleDraw function returns immediately and the drawing is ended. A client could also use the OLE\_QUERY\_PAINT notification to take some actions within the callback function and then return TRUE to indicate that drawing should continue. Any actions the client takes at this time should not interfere with the drawing operation; for example, the client should not scroll the window.

If the target device for an object changes (for example, when the user changes printers), the client application should call the OleSetTargetDevice function. The client should also call OleSetTargetDevice whenever an object is created or loaded.

If the size of the presentation rectangle for the object changes (for example, through action by the user) the client application should call the OleSetBounds function. After calling OleSetBounds, the client should call the OleUpdate function to update the object and then OleDraw to redisplay it.

### Opening and Closing Objects

When the user requests the client application to activate an object, the client should check whether the object is busy by calling the OleQueryReleaseStatus function. If the object is busy, the client should either refuse the request to open the object or enter a message-dispatch loop, waiting for the OLE\_RELEASE notification.

If the object to be activated is not busy, the client should call the OleActivate function. The library notifies the client when the server is open or when an error occurs.

The OleActivate function allows the client application to specify whether to display the activated object in a window of the server application. A client might hide the server window if an object is updated automatically.

A client application can use the OleQueryOpen function to determine whether a specified object is open. The OleClose function allows the client to close an open object. Closing an object terminates the connection with the server. To reestablish a terminated connection between a linked object and an open server, the client can use the OleReconnect function. To close an open object and release it from memory, a client application can call the OleRelease function.

The first time a client application activates a particular embedded object, the client should call the OleSetHostNames function, specifying the string the server window should display in its title bar. This string should be the name of the client document containing the object. The client does not need to call OleSetHostNames every time an embedded object is activated, because the library maintains a record of the specified names.

### Deleting Objects

To permanently delete an object from a document, the client should call the OleDelete function. OleDelete closes the specified object, if necessary, before deleting it.

### Client Cut and Copy Commands

A client application can copy an object to the clipboard by simply opening the clipboard, calling the OleCopyToClipboard function, and closing the clipboard again. If the client supports delayed rendering, however, it should follow these steps to cut or copy an object to the clipboard:

- 1 Open and empty the clipboard.
- 2 Put the preferred data formats on the clipboard.
- 3 Call the OleEnumFormats function to retrieve the formats for the object.

- 4 Call the SetClipboardData function to put the formats on the clipboard, specifying NULL for the handle of the data.

If the call to the OleEnumFormats function retrieves the ObjectLink format, the client should call SetClipboardData with OwnerLink instead of ObjectLink format. (For more information, see the following description of the OleCopyToClipboard function.)

- 5 Put any additional presentation data formats on the clipboard.
- 6 Close the clipboard.

To support the Cut command on the Edit menu, an application can call OleCopyToClipboard and then delete the object by using the OleDelete function. (The client can put only one of the selected objects on the clipboard, even when the user has selected and cut or copied multiple objects. In this case, the client typically puts the first object in the selection onto the clipboard.)

The OleCopyToClipboard function always copies OwnerLink format, not ObjectLink format, to the clipboard. For embedded objects, Native data always precedes the OwnerLink format. If a linked object uses Native data, OwnerLink format always precedes the Native data. If an application uses the OleGetData function to retrieve data from a linked object that has been copied by using OleCopyToClipboard, it should specify ObjectLink format, not OwnerLink format, even if OwnerLink format was put on the clipboard.

When an application that can act as both a client and server copies a selection to the clipboard that contains one or more objects, it should first allocate enough memory for the selection. To discover how much memory is required for each object, the application can call the OleQuerySize function. When memory has been allocated, the application should call the OleRegisterClientDoc function, specifying Clipboard for the document name. (In this case, the handle returned by the call to OleRegisterClientDoc identifies a document that is used only during the copy operation.) To save each object to memory, the application calls the OleClone function, calls the OleSaveToStream function for the cloned object, and then calls the OleRelease function to free the memory for the cloned object. When the selection has been saved to the stream, the application can call the SetClipboardData function. If SetClipboardData is successful, the application should call the OleSavedClientDoc function. The application then calls the OleRevokeClientDoc function, specifying the handle retrieved by the call to OleRegisterClientDoc.

For more information about the Cut and Copy commands, see Server Cut and Copy Commands.

### Creating Objects

A client application can put linked and embedded objects in a document by pasting them from the clipboard, creating them from a file, copying them from other objects, or by starting a server application to create them directly.

### Object-Creation Functions

Each of the following functions creates an embedded or linked object in a specified document:

Function	Description
<u>OleClone</u>	Creates an exact copy of an object.
<u>OleCopyFromLink</u>	Creates an embedded object that is a copy of a linked object.
<u>OleCreate</u>	Creates an embedded object of a specified class.
<u>OleCreateFromClip</u>	Creates an object from the clipboard. This function typically creates an embedded object.
<u>OleCreateFromFile</u>	Creates an object by using the contents of a file. This function typically creates an embedded object.
<u>OleCreateFromTemplate</u>	Creates an embedded object by using another object as a template.
<u>OleCreateInvisible</u>	Creates an object without displaying the server application to the user.
<u>OleCreateLinkFromClip</u>	Creates an object by using information on the clipboard. This function typically creates a linked object.
<u>OleCreateLinkFromFile</u>	Creates an object by using the contents of a file. This function typically creates a linked object.
<u>OleObjectConvert</u>	Creates an object that supports a specified protocol by converting an existing object.

Each of these functions requires a parameter that points to an OLEOBJECT structure when the function returns. Server applications often create an OLEOBJECT structure whenever an object is created; OLEOBJECT points to functions that describe how the server interacts with the object. Before the client library gives the client application a pointer to this structure, the library includes with the structure some

internal information corresponding to the OwnerLink or ObjectLink data. This internal information allows the client library to identify the correct server when an OLE function such as OleActivate passes it a pointer to an OLEOBJECT structure. For more information about the OLEOBJECT structure, see Starting a Server Application.

Each new object must have a name that is unique to the client document. Although meaningful object names can be helpful, some applications assign unique object names simply by incrementing a counter for each new object. For more information about object names, see Document Management.

If a client application implements the Insert Object command, it should use the registration database to find out what OLE servers are available and then list those servers for the user. When the user selects one of the servers and chooses the OK button, the client can use the OleCreate function to create an object at the current position.

The OleCopyFromLink, OleCreate, and OleCreateFromTemplate functions always create an embedded object. The other object-creation functions can create either an embedded object or a linked object, depending on the order and type of available data.

If a client application's callback function receives the OLE\_RELEASE notification after the client calls the OleCreate or OleCreateFromFile function, the client should respond by calling the OleQueryReleaseError function. If OleQueryReleaseError shows that there was an error when the object was created, the client application should delete the object.

Whenever an object-creation function returns OLE\_WAIT\_FOR\_RELEASE, the calling application should either wait for the OLE\_RELEASE notification or notify the user that the object cannot be created. For more information, see Asynchronous Operations.

If a client application accepts files dropped from File Manager, it should respond to the WM\_DROPFILES message by calling the OleCreateFromFile function and specifying Packager for the lpszClass parameter.

#### Paste and Paste Link Commands

A client application should follow these steps to create an embedded or linked object by pasting from the clipboard:

- 1 Call the OleQueryCreateFromClip function to determine whether to enable the Paste command. If this function fails when StdFileEditing is specified for the lpszProtocol parameter, call it again, specifying Static.
- 2 Call the OleQueryLinkFromClip function to determine whether to enable the Paste Link command.
  - If the user chooses the Paste command, open the clipboard and call the OleCreateFromClip function.
  - If the user chooses Paste Link, open the clipboard and call the OleCreateLinkFromClip function.
- 3 Close the clipboard.
- 4 Call the OleQueryType function to determine the kind of object created by the creation function. (Depending on the order of clipboard data, OleCreateFromClip can sometimes create a linked object and OleCreateLinkFromClip can sometimes create an embedded object.)

The client application should put the pasted data or object into the document at the current position. The client should select the object so that the user can work with it immediately. If both the OleQueryCreateFromClip and OleQueryLinkFromClip functions fail but there is data on the clipboard that the client can interpret, the client should enable the Paste command.

If the information on the clipboard is incomplete--for example, if Native data is not accompanied by the OwnerLink format--the Paste command should insert a static object into the document. (A static object consists of the presentation data for an object; it cannot be edited by using standard OLE techniques. Attempts to open static objects fail and generate no notifications.)

If the client application implements the Paste Special command, it should use the EnumClipboardFormats function to produce a list of data formats on the clipboard. The client should also check the registration database to find the full name of the server application. The Paste Link button in the Paste Special dialog box works in exactly the same way as the Paste Link command on the Edit menu.

If the DDE Link format is available on the clipboard instead of ObjectLink format, the client application should perform the same link operation that it supported prior to the implementation of OLE.

#### Undo Command

A client application can use the OleClone function to support the Undo command. A cloned object is identical to the original except for connections to the server application; the cloned object is not automatically connected to the server. When the server is closed and the object is updated, the saved

copy of the object gives the user the opportunity to undo all of the changes made in the server. Support for the Undo command is provided by the client application, because the server cannot maintain a record of the prior states of objects.

The Undo command restores an object to its condition prior to the last update from the server. To support this behavior, the client application must clone the object when it is first activated and then clone the updated object when an update occurs; the client must maintain two clones of the object. The clone of the original object must be maintained so that an updated object can be restored if the user chooses the Undo command. The clone of the updated object must be maintained to support the Undo command if the updated object is updated again. Because the data changes when the update occurs, the clone for supporting the Undo command must be made before any updates occur.

Because the client application cannot distinguish between different types of object activation, the client must clone an object for verbs that do not edit the object, even though no updates can occur in those cases.

#### Class Name Object Command

A client application can implement the Class Name Object command by using the OleActivate function. OleActivate includes a parameter that allows the client to specify the verb chosen by the user.

#### Links Command

When a user chooses the Links command, a dialog box appears listing every linked object in the document. The selected links are highlighted in the dialog box. The dialog box makes it possible for the user to invoke the verbs for an object, select whether link updating should be automatic or manual, update a link immediately, cancel a link, and repair broken links. For more information about this dialog box, see Microsoft Windows User Interface Guidelines.

The Links dialog box includes buttons that allow the user to activate the primary and secondary verbs for an object. A client application can implement these buttons by using the OleActivate function.

A client application can use the OleGetLinkUpdateOptions and OleSetLinkUpdateOptions functions to support the link-update radio buttons in the Links dialog box. The following are the three possible update options:

Option	Description
oleupdate_always	Update the linked object whenever possible. This option supports the Automatic link-update radio button in the Links dialog box.
oleupdate_onsave	Update the linked object when the source document is saved by the server.
oleupdate_onsave	Update the linked object only on request from the client application. This option supports the Manual link-update radio button in the Links dialog box.

These update options control when updates to the presentation of an object occur. The contents of the source document are used to update the presentation whenever the link is activated.

To support the Update Now button in the Links dialog box, an application can call the OleUpdate function. When a user chooses Update Now, the client application should update the links the user selected.

A user's choosing the Cancel Link button in the Links dialog box changes an object into a picture that an application cannot edit by using standard OLE techniques. An application can implement the Cancel Link button by using the OleObjectConvert function.

A client application should activate the Change Link button in the Links dialog box only if all the selected links are to the same source document. When the client has the correct information, it can repair the link by using the OleGetData and OleSetData functions. To retrieve the link information for an object, a client can call the OleGetData function, specifying the ObjectLink format. (The call to OleGetData fails if ObjectLink is specified and the object is not a link.) A client can retrieve class information by using OleGetData and specifying either the OwnerLink format (for embedded objects) or the ObjectLink format (for linked objects). The client can make it possible for the user to edit the link information and store it in the object by using the OleSetData function, specifying the ObjectLink format.

#### Closing a Client Application

A client application should use the OleRelease function to remove all objects from memory when it shuts down. If the library returns the OLE\_WAIT\_FOR\_RELEASE value instead of OLE\_OK, the client should not quit. The client can perform many cleanup tasks while waiting for the OLE\_RELEASE notification--for example, it can close files, free memory, and hide windows.

The OLE\_RELEASE notification to the client's callback function indicates that an operation has finished in a server application, but it does not identify the operation or indicate whether the operation



was successful. A client application can call the OleQueryReleaseStatus function to determine whether an operation has been completed for a specified object. The OleQueryReleaseMethod function indicates the nature of the operation that has finished for a specified object. To discover the error value for the operation, the client can call the OleQueryReleaseError function.

If a client owns the clipboard when it quits, it should make sure that the data on the clipboard is complete and in the correct order.

## Server Applications

### Server Applications

An OLE server supplies functions that the server library calls when a user works with an object. The server library, OLESVR.DLL, uses DDE commands to communicate with the client library. When the client application calls one of the functions in the OLE API, the client library informs the server library and the server library routes the request to the appropriate function in the server-supplied list of function pointers.

In addition to the specialized functions that the server creates and which are called by the server library, there are ten OLE functions that allow a server to control the library's ability to gain access to the server and the documents and objects it controls:

Function	Description
<u>OleBlockServer</u>	Queues requests to the server until the server calls the <u>OleUnblockServer</u> function.
<u>OleRegisterServer</u>	Registers the specified server with the library. Information registered includes the class name and instance and whether the server supports single or multiple instances.
<u>OleRegisterServerDoc</u>	Registers a document with the server library.
<u>OleRenameServerDoc</u>	Renames the specified document.
<u>OleRevertServerDoc</u>	Restores a document to a previously saved state, without closing the document.
<u>OleRevokeObject</u>	Revokes access to the specified object.
<u>OleRevokeServer</u>	Revokes access to the specified server, closing any documents and ending communication with client applications.
<u>OleRevokeServerDoc</u>	Revokes access to the specified document.
<u>OleSavedServerDoc</u>	Informs the library that a document has been saved. Calling this function is equivalent to sending the OLE_SAVED notification.
<u>OleUnblockServer</u>	Processes a request from a queue created when the server application called the <u>OleBlockServer</u> function.

The OleRevokeServer and OleRevokeServerDoc functions can return OLE\_WAIT\_FOR\_RELEASE. When a server application receives this error value, it should take the same action as a client application, dispatching messages until the server library calls the corresponding Release function.

### Starting a Server Application

When a server application starts, it should follow these steps:

- 1 Register window classes and window procedures for the main window, documents, and objects.
- 2 Initialize the function tables for the OLESERVERVTBL, OLESERVERDOCVTBL, and OLEOBJECTVTBL structures.
- 3 Register the clipboard formats.
- 4 Allocate memory for the OLESERVER structure.
- 5 Register the server with the library by calling the OleRegisterServer function.
- 6 Check for the /Embedding and /Embedding filename options on the command line and act according to the following guidelines. (Applications should also check for -Embedding whenever they check for these options.)
  - If neither /Embedding nor /Embedding filename is present, call the OleRegisterServerDoc function, specifying an untitled document.
  - If the /Embedding option is present, do not register a document or display a window. (In this case, the server takes actions only in response to calls from the server library.)
  - If the /Embedding filename option is present, do not display a window. Process the filename string and call the OleRegisterServerDoc function.

The OLESERVERVTBL, OLESERVERDOCVTBL, and OLEOBJECTVTBL structures are tables of function pointers. The server library uses these structures to route requests from the client application to the server. The server application should create the function pointers in these structures by using the MakeProcInstance function. The functions should also be exported in the application's module-definition file.

The OLESERVER structure contains a pointer to an OLESERVERVTBL structure. The OLESERVERVTBL structure contains pointers to functions that control such fundamental server tasks as opening files, creating objects, and terminating after an editing session. Several of the functions pointed to by the OLESERVERVTBL structure cause the server to allocate and initialize an OLESERVERDOC structure.

The OLESERVERDOC structure contains a pointer to an OLESERVERDOCVTBL structure. The OLESERVERDOCVTBL structure contains pointers to functions that control such tasks as saving or closing documents or setting document dimensions. The OLESERVERDOCVTBL structure also contains a function that causes the server to allocate and initialize an OLEOBJECT structure.

The OLEOBJECT structure contains a pointer to an OLEOBJECTVTBL structure. The OLEOBJECTVTBL structure contains pointers to functions that operate on objects. After the server application creates an OLEOBJECT structure, the server library gives information about the structure to the client library. The client library then creates a parallel OLEOBJECT structure (including internal information identifying the server application, the document, and the item for the object) and passes a pointer to that structure to the client application.

This hierarchy of structures--OLESERVER, OLESERVERDOC, and OLEOBJECT--makes it possible for a server to open as many documents as the library requests and for each document to contain as many objects as necessary.

A server application can register the clipboard formats by calling the RegisterClipboardFormat function for each format, specifying Native, OwnerLink, ObjectLink, and any other formats it requires.

When the server application starts, it creates an OLESERVER structure and then registers it with the library by calling the OleRegisterServer function. When this function returns, one of its parameters points to a server handle. The library uses this handle of refer to the server, and the server uses it in calls to the server-specific OLE functions.

If an OLE server application is also a DDE server, the class name specified in the call to the OleRegisterServer function cannot be the same as the name of the executable file for the application.

When a client working with a compound document opens a linked or embedded object for editing, the client library starts the server using the /Embedding command-line option. The server uses this option to determine whether the object has been opened directly by a user or as part of an editing session for linked and embedded objects. (If the object is a linked object, the /Embedding option is followed by a filename.) When a server is started for an embedded object with the /Embedding option, the server should not create a document or show a window. Instead, it should call the OleRegisterServer function and then enter a message-dispatch loop. (If the server is started with the /Embedding filename option, it should also call the OleRegisterServerDoc function.) The server then takes actions in response to calls from the library. The server should not make itself visible until the library calls the Show or DoVerb function in the OLEOBJECTVTBL structure. (Server applications should check for both -Embedding and /Embedding.)

By calling the OleBlockServer function, a server application can cause requests from the client library to be saved in a queue. When the server is ready for the server library to process the requests, it can call the OleUnblockServer function. It is best to use the OleUnblockServer function prior to the GetMessage function in a message loop, so that all blocked requests are unblocked before getting the next message. (Often a server returns OLE\_BUSY instead of calling OleBlockServer. Returning OLE\_BUSY has two advantages: It allows the client to decide whether to retry the message or discontinue the operation, and it allows the server to choose which requests to process.)

When an error occurs in a server-supplied function, the server should return the OLESTATUS error value that best describes the error. The OLE libraries use these error values to help determine the appropriate behavior in error situations. However, the client application does not necessarily receive the error values the server returns; the OLE libraries may change error values before passing them to the client application.

#### Opening a Document or Object

Whenever the server library calls the Open, Create, CreateFromTemplate, or Edit function in the OLESERVERVTBL structure, the server creates an OLESERVERDOC structure. If the document is opened by a call from the server library, the server application returns the OLESERVERDOC structure to the library. If the document is opened directly by a user, however, the server should call the OleRegisterServerDoc function to register the document with the library. The library then uses the GetObject function in the OLESERVERDOCVTBL structure to request the server to create an OLEOBJECT structure for each object requested by the client application.

A new instance of the server application is typically started when the client activates a linked or embedded object. This new instance is unnecessary if the object is already open in an instance of the server or if the server is a single-instance (MDI) server that is already open. For more information about

the rules for starting new instances of server applications, see Microsoft Windows User Interface Guidelines.

Whether the server library starts a new instance of a server to edit an embedded or linked object depends upon the value specified when the server calls the OleRegisterServer function.

#### Server Cut and Copy Commands

A server application should follow these steps to cut or copy onto the clipboard data that a client can then use to create an embedded or linked object:

- 1 Open and empty the clipboard.
- 2 Put the data formats that describe the selection on the clipboard, using the SetClipboardData function.
- 3 Close the clipboard.

If the server cuts data onto the clipboard, rather than copying it, the server typically does not offer ObjectLink or Link formats, because the source for the data has been removed from the document.

The server should put data on the clipboard in the order given in Clipboard Conventions.

Typically, the server puts server-specific formats, Native format, OwnerLink format, and presentation formats on the clipboard. If it can support links, the server also puts ObjectLink format and, when appropriate, Link format on the clipboard. The server must provide a presentation format (CF\_METAFILE, CF\_BITMAP, or CF\_DIB) if the server does not have an object handler. Native data can be used as a presentation format only if the server has an object handler that can use the Native data.

If a user copies onto the clipboard a selection that includes an embedded object or a link, the data formats the server should copy depend upon whether the container document modifies the object or link. If the document does not modify the object or link, the best formats are the Native and OwnerLink formats from the original source of the object. If the document modifies the object or link--for example, by recoloring it--the best formats are the Native and OwnerLink formats from the container document.

If a server uses a metafile as the presentation format for an object, the mapping mode for that metafile must be MM\_ANISOTROPIC. When a server application uses fonts in these metafiles, it can improve performance by using TrueType fonts. (Metafiles scale better when they use TrueType fonts.) To use TrueType fonts exclusively, the server should set bit 2 (04h) of the lpPitchandFamily member of the LOGFONT structure.

The OLE libraries express the size of every object in MM\_HIMETRIC units. Neither the width nor height of an object should exceed 32,767 in MM\_HIMETRIC units.

#### Update, Save As, and New Commands

When a server is started as part of editing an object from within a compound document, the server application should change the Save command on the File menu to Update. When the user chooses the Update command, the server should call the OleSavedServerDoc function.

When the user chooses the Save As, New, or Open command in a single-document server, the application should display a message asking the user whether to update the object in the compound document before performing the action. When the user chooses the Save As command, the server should call the OleRenameServerDoc function. If the user responds to the message by choosing to save changes in the object before renaming the document, the server should call the OleSavedServerDoc function before calling OleRenameServerDoc. For embedded objects, choosing the Save As command causes the connection with the client to be broken, because this command reassociates a document in memory with the specified new file. For linked objects, calling OleRenameServerDoc when the user chooses Save As makes it possible for the client to associate the link with the new file.

Most server applications maintain a "dirty" flag that records whether changes have been made to each open document in an instance. The following table shows the rules that apply to this flag when the server edits an embedded object. By following these rules, a server can ensure that this flag is TRUE when the document being edited in the server matches the embedded object in the client and that, otherwise, this flag is FALSE.

Flag	Condition
TRUE	Library calls the Create function in the <u>OLESERVERVTBL</u> structure.
TRUE	Library calls the <u>CreateFromTemplate</u> function in <u>OLESERVERVTBL</u> .
TRUE	Document is changed in server.
FALSE	Library calls the Edit function in <u>OLESERVERVTBL</u> .

FALSE      Library calls the GetData function in OLEOBJECTVTBL with the Native data format. (The flag should not change for any other formats.)

A server following these rules displays the message asking whether to update the object whenever it destroys a document that was editing an embedded object and the "dirty" flag is TRUE.

In an MDI server application, the New and Open commands on the File menu simply open a new window, and the connection with the client application remains unchanged. The user can continue to work with the server application after choosing one of these commands, but when the user exits the server application, the focus does not necessarily return to the client application.

Typically, a server can call the OleSavedServerDoc function whenever an object needs to be updated in the client document, including when the server closes the document. When the server closes the document and the object should be updated, the server sends the OLE\_CLOSED notification. Client applications receive the OLE\_CLOSED notification for embedded objects but not for linked objects, because the server library intercepts the notification for linked objects.

#### Closing a Server Application

The server library calls the Exit function in the OLESERVERVTBL structure when the server must quit. The server library calls the Release function to inform the server that it is safe to quit; the server does not necessarily stop when the library calls Release.

The server must exit when it is invisible and the library calls Release. (The only exception is when an application supports multiple servers; in this case, an invisible server is sometimes not revocable when the library calls Release.) If the server has no open documents and it was started with the /Embedding option (indicating that it was started by a client application), the server should exit when the library calls the Release function. If the user explicitly loads a document into a single-instance (MDI) server, however, the server should not exit when the library calls Release.

When the user closes a server that has edited an embedded object without updating changes to the client application, the server should display a message asking whether to save the changes. If the user chooses to save the changes, the server should send the OLE\_CLOSED notification and call the OleRevokeServerDoc function. (Because sending OLE\_CLOSED prompts the server library to send data to the client library, it is not necessary to send OLE\_CHANGED or OLE\_SAVED. If the user chooses not to save the changes, the server should simply call the OleRevokeServerDoc function (without sending OLE\_CLOSED).

A server can use the OleRevokeObject function to revoke a client's access to an object--for example, if the user destroys the object. Similarly, the OleRevokeServerDoc function revokes a client's access to a document. (Because OleRevokeServerDoc revokes a client's access to all objects in a document, an application that uses OleRevokeServerDoc does not need to call the OleRevokeObject function for objects in that document.) To terminate all conversations with client applications, the server can call the OleRevokeServer function. These functions inform the server library that the specified items are no longer available.

A server application can receive OLE\_WAIT\_FOR\_RELEASE--for example, the OleRevokeServerDoc function can return this value. Although a server can enter a message-dispatch loop and wait for the library to call the server's Release function, servers should never enter message-dispatch loops inside any of the server-supplied functions that are called by the server library.

The client application should not instruct the server to close the document or exit when the server is editing a linked object, unless the server is updating the link without displaying the object to the user. Because a linked object exists independently of the client, the user controls saving and closing the document by using the server application.

If a server application owns the clipboard when it closes, it should make sure that the data on the clipboard is complete and in the correct order. For example, any Native data should be accompanied by the OwnerLink format.

## Object Handlers

### Object Handlers

An application developer can use object handlers to introduce customized features into implementations of linked and embedded objects. When an object handler exists for a class of object, the object handler supplants some or all of the functionality that is usually provided by the client library and the server application. The object handler can take specialized action for any of the functions it intercepts. The object handler passes functions that it does not take action on to the client library, which then implements the default processing for that class.

An application might use an object handler to render Native data as the presentation data for an object, instead of using metafiles or bitmaps. Object handlers could also be used to implement special behavior when an object is opened.

### Implementing Object Handlers

A server installing an object handler registers the handler with the registration database, using the keyword handler. Whenever a client application calls one of the object-creation functions, the client library uses the class name specified for the object and the handler keyword to search the registration database. If the library finds an object handler, the client library loads the handler and calls it to create the object. The handler can create an object for which all of the creation functions and methods are defined by the handler, or it can call default object-creation functions in the client library.

The client library exports the object-creation OLE functions with new names; in each case, the prefix "Ole" is changed to "Def" (for "default"). Object handlers can import any of these functions and use them when creating objects.

Object handlers must import the following functions:

OLE function	Name exported by client library
<u>OleCreate</u>	DefCreate
<u>OleCreateFromClip</u>	DefCreateFromClip
<u>OleCreateFromFile</u>	DefCreateFromFile
<u>OleCreateFromTemplate</u>	DefCreateFromTemplate
<u>OleCreateLinkFromClip</u>	DefCreateLinkFromClip
<u>OleCreateLinkFromFile</u>	DefCreateLinkFromFile
<u>OleLoadFromStream</u>	DefLoadFromStream

When an object handler defines a function that is to be called by the client application, it should use the same name as the corresponding OLE function the client calls, with the prefix "Ole" replaced by "Dll". For example, when an object handler uses the DefCreate function exported by the client library, the handler should use it inside a function named DllCreate. When the client library finds an object handler for a class of object, it calls handler-specific object-creation functions by specifying this "Dll" prefix.

When the handler calls one of the default object-creation functions, it receives a handle of an OLEOBJECT structure, which in turn points to the OLEOBJECTVTBL structure containing the current object-management functions. The object handler should copy this OLEOBJECTVTBL structure and customize the structure by replacing any function pointers in the structure with pointers to functions of its own. (If the object handler saves the pointers to the default functions, any of the replacement functions can also call the default functions in the table of function pointers.) When the object handler has finished customizing the structure, it should replace the pointer to the old OLEOBJECTVTBL structure with a pointer to the modified OLEOBJECTVTBL structure.

When the client makes a call to a function in the client library, the call is dispatched through the object handler's OLEOBJECTVTBL structure. If the object handler has replaced the function pointer, the call is routed to the function supplied by the handler. Otherwise, the call is routed to the client library.

Each OLECLIENT, OLEOBJECT, OLESERVER, OLESERVERDOC, or OLESTREAM structure contains a pointer to a structure that contains a table of function pointers. (Structures containing tables of function pointers are identified with the "VTBL" suffix.) Each of the structures containing a pointer to a "VTBL" structure can also contain extra instance-specific information. This information is meaningful only to the application that supplies it and should not be used by other applications; for example, an object handler should not attempt to use any instance-specific information in an OLECLIENT structure.

The object handler should use the "Def" and "Dll" renaming conventions when it defines specialized functions. For example, if an object handler modifies the Draw function from an object's

OLEOBJECTVTBL structure, it should copy that Draw function to a function named DefDraw and replace the Draw function with a specialized function named DllDraw. Inside the DllDraw function, the object handler can call DefDraw if the default drawing operation is appropriate in a particular case.

The following example demonstrates this process of copying and replacing pointers to functions. Functions with the "Dll" prefix should be exported in the module-definition file.

```

/* Declare the DllDraw and DefDraw functions. */

OLESTATUS FAR PASCAL DllDraw(LPOLEOBJECT, HDC, LPRECT, LPRECT, HDC);
OLESTATUS (FAR PASCAL *DefDraw)(LPOLEOBJECT, HDC, LPRECT, LPRECT, HDC)
;

/* Copy the Draw function from OLEOBJECTVTBL to DefDraw. */

    DefDraw    = lpobj->lpvtbl->Draw;

/* Copy DllDraw to OLEOBJECTVTBL. */

    *lpobj->lpvtbl->Draw = DllDraw;

OLESTATUS FAR PASCAL DllDraw(lpObject, hdc, lpBounds, lpWBounds,
    hdcFormat)
LPOLEOBJECT lpObject;
HDC    hdc;
LPRECT lpBounds;
LPRECT lpWBounds;
HDC    hdcFormat;
{
    /* Return DefDraw if Native data is not available. */

    if ((*lpobj->lpvtbl->GetData) (lpobj, cfNative, &hData) != OLE_OK)
    return (*DefDraw) (lpobj, hdc, lpBounds, lpWBounds, hdcFormat);
    .
    .
    .
}

```

#### Creating Objects in an Object Handler

Most of the object-creation functions in the OLE API work in exactly the same way when they are renamed and used by object-handler DLLs. Two functions are somewhat different, however:

OleCreateFromClip and OleLoadFromStream.

#### DefCreateFromClip and DllCreateFromClip

When the client library calls the DllCreateFromClip function, the library includes a parameter that is not specified in the original call to the OleCreateFromClip function. This parameter, objtype, specifies whether the object being created is an embedded object or a link; its value can be either OT\_LINK or OT\_EMBEDDED.

The following syntax block shows the objtype parameter when an object handler uses the DefCreateFromClip function. The DllCreateFromClip function has exactly the same syntax as DefCreateFromClip. For a full description of all the parameters, see the description of the OleCreateFromClip function.

```

OLESTATUS DefCreateFromClip(lpszProtocol, lpclient, lhclientdoc, lpszObjname, lplpobject,
    renderopt, cfFormat, objtype);
LPSTR lpszProtocol; /* address of string for protocol name */
LPOLECLIENT lpclient;
/* address of client structure
/
LHCLIENTDOC lhclientdoc;
/* long handle of client document
/

```

```

LPSTR lpszObjname;
* string for object name
/
LPOLEOBJECT FAR * lplpobject;
* address of pointer to object
/
OLEOPT_RENDER renderopt;
* rendering options
/
OLECLIPFORMAT cfFormat;
* clipboard format
/
LONG objtype;
* OT_LINKED or OT_EMBEDDED
/

```

If `DllCreateFromClip` calls `DefCreateFromClip`, `DllCreateFromClip` should pass it the `objtype` parameter along with the other parameters from the version of `DefCreateFromClip` that was exported by the client library. `DllCreateFromClip` can modify some of these parameters before passing them back to `DefCreateFromClip`. For example, the object handler could specify a different value for the `renderopt` parameter when it calls `DefCreateFromClip`. If the client calls this function with `olerender_draw` for `renderopt` and the handler performs the drawing with Native data, the handler could change `olerender_draw` to `olerender_none`. If the client calls this function with `olerender_draw` for `renderopt` and the handler calls the `GetData` function and performs the drawing based on a class-specific format, the handler could change `olerender_draw` to `olerender_format`. If the handler needed a different rendering format than the format specified by the client application, the object handler could also change the value of the `cfFormat` parameter in the call to `DefCreateFromClip`.

If an object handler uses Native data to render an embedded object, the handler can call the library and specify `olerender_none`. If a handler uses Native data to render a linked object, it can use `olerender_format` and specify Native data. When the handler's `Draw` function is called, the handler calls the `GetData` function, specifying Native data, to do the rendering. If a handler uses a private data format, the procedure is the same--except that the private format is specified with the `olerender_format` option and with the `GetData` function.

#### DefLoadFromStream and DllLoadFromStream

When the client library calls the `DllLoadFromStream` function, the library includes three parameters that are not specified in the original call to the `OleLoadFromStream` function. One of the additional parameters is `objtype`, as described for `DefCreateFromClip` and `DllCreateFromClip`. The other two parameters are `aClass`, which is an atom containing the class name for the object, and `cfFormat`, which specifies a private clipboard format that the object handler can use for rendering the object.

The following syntax block shows the `objtype`, `aClass`, and `cfFormat` parameters when an object handler uses the `DefLoadFromStream` function. The `DllLoadFromStream` function has exactly the same syntax as `DefLoadFromStream`. For a full description of all the parameters, see the description of the `OleLoadFromStream` function in the Microsoft Windows Programmer's Reference, Volume 2.

```

OLESTATUS DefLoadFromStream(lpstream, lpszProtocol, lpclient,
    lhclientdoc, lpszObjname, lplpobject, objtype, aClass, cfFormat);
LPOLESTREAM lpstream; /* address of stream for object */
LPSTR lpszProtocol; /* address of string for protocol name */
LPOLECLIENT lpclient; /* address of client structure */
LHCLIENTDOC lhclientdoc; /* long handle of client document */
LPSTR lpszObjname; /* string for object name */
LPOLEOBJECT FAR * lplpobject; /* address of pointer to object */
LONG objtype; /* OT_LINKED or OT_EMBEDDED */
ATOM aClass; /* atom containing object's class name */
OLECLIPFORMAT cfFormat; /* private data format for rendering */

```

If `DllLoadFromStream` calls `DefLoadFromStream`, `DllLoadFromStream` should pass it the three additional parameters along with the other parameters from the version of `DefLoadFromStream` that was exported by the client library. `DllLoadFromStream` can modify some of these parameters before passing them back to `DefLoadFromStream`. For example, the object handler could modify the value of the `cfFormat` parameter to specify a private data format it would use to render the object.



When the client calls the object handler with DefLoadFromStream, the handler uses the Get function from the OLESTREAMVTBL structure to obtain the data for the object.

## Direct Use of Dynamic Data Exchange

The OLE libraries, OLECLI.DLL and OLESVR.DLL, use DDE messages to communicate with each other. Although client and server applications can use DDE directly, without employing OLECLI.DLL or OLESVR.DLL, this method of implementing OLE is not recommended. Future enhancements to the OLE libraries will benefit applications that use the libraries but will not benefit applications that use DDE directly.

The following information about the DDE-based OLE protocol is provided for applications that must implement DDE directly, despite losing the ability to take advantage of future enhancements to the system.

Implementation of the OLE protocol requires implementation of the underlying DDE protocol. All the standard DDE rules and facilities apply. Applications that conform to this protocol must also conform to the DDE specification. Conforming to this specification implies supporting the System topic and the standard items in that topic.

### Client Applications and Direct Use of Dynamic Data Exchange

When opening a link or an embedded document, the client application should look up the class name in the registration database, as described in Registration.

The following pseudocode illustrates the chain of events for a client implementing OLE through DDE. Whenever a client that attempts to establish a conversation with a server receives responses from more than one server, the client should accept the first server and reject the others.

Linked object:

```
WM_DDE_INITIATE class name, document name
if not found {
WM_DDE_INITIATE class name, OLESystem
if not found {
WM_DDE_INITIATE class name, System
if not found {
launch application name, /Embedding
fLaunched = true
WM_DDE_INITIATE class name, OLESystem
if not found {
WM_DDE_INITIATE class name, System
if not found
return error
}
}
}

/*
* Now there is a conversation with the server on the System or
* OLESystem topic.
*/

WM_DDE_EXECUTE StdOpenDocument(DocumentName)
WM_DDE_INITIATE class name, document name
if not found {
if(fLaunched) WM_DDE_EXECUTE StdExit /* clean up */
return error
}
}

/*
* Now there is a conversation with the correct document.
*/
```

Embedded object:

```
WM_DDE_INITIATE class name, OLESystem
if not found {
WM_DDE_INITIATE class name, System
```

```

if not found {
    launch application name, /Embedding
    fLaunched = true
    WM_DDE_INITIATE class name, OLESystem
    if not found {
        WM_DDE_INITIATE class name, System
    }
    if not found
return error
}
}
}

/*
 * Now there is a conversation with the server on the system or
 * OLESystem topic.
 */

DDE_EXECUTE StdEditDocument(DocumentName)

/*
 * Or StdCreateDoc if this is an Insert Object command
 */

WM_DDE_INITIATE class name, document name
if not found {
if(fLaunched) DDE_EXECUTE StdExit/* clean up */
return error
}

/* Now there is a conversation with the correct document. */

```

#### Server Applications and Direct Use of Dynamic Data Exchange

When a server receives the /Embedding command-line argument, it should not create a new default document. Instead, it should wait until the client sends either the StdOpenDocument command or the StdEditDocument command followed by the Native data and then instructs the server to show the window. The server can use the StdHostNames item to display the client's name in the window title.

The following pseudocode illustrates the chain of events for a server implementing OLE through DDE. The example shows two cases: one in which the server reuses a single instance for editing all objects (in MDI child windows), and another in which a new instance is used for each object. Applications that use a new instance for each object should reject requests to open or create a new document when they already have a document open.

MDI application:

```

case WM_DDE_INITIATE:
if class name == this class {
if (DocumentName == OLESystem || DocumentName == System)
    WM_DDE_ACK
else if DocumentName == name of some open document
    WM_DDE_ACK
}

```

Multiple-instance application:

```

case WM_DDE_INITIATE:
if class name == this class {
if (DocumentName == OLESystem || DocumentName == System) {
    if no documents are open
        WM_DDE_ACK
}
else if DocumentName == name of some open document
    WM_DDE_ACK
}
}

```

## Conversations

Document operations are performed during conversations with an application's OLESystem or System topic. The document's class name is used to establish the conversation.

Data transfer and negotiation operations are performed during conversations with the document (that is, the topic). The document name is used to establish the conversation.

Note that the topic name is used only in initiating conversations and is not fixed throughout the conversation; permitting the document to be renamed does not mean that there will be two names. Therefore, it is reasonable to tie the topic name to the document name.

## Items for the System Topic

An application using DDE-based OLE can use three new items for the System topic: the Topics item, the Protocols item, and the Status item.

The Topics item returns a list of DDE topic names that the server application has open. Where topics correspond to documents, the topic name is the document name.

The Protocols item returns a list of protocol names supported by the application. The list is returned in tab-separated text format. A protocol is a defined set of DDE execute strings and item and format conventions that the application understands. The protocol currently defined for linked and embedded objects is the following:

Protocol: StdFileEditing commands/items/formats

For compatibility with client applications that were written before the implementation of the OLE protocol, server applications that use the DDE protocol directly should also include the string Embedding in the list of protocols.

The Status item is a text item that returns Ready if the server is prepared to respond to DDE requests; otherwise, it returns Busy. This item can be queried to determine if the client should offer such functions as one that gives the user an opportunity to update the object. Because it is possible that a server could reject or defer a request even if Status returns Ready, client applications should not depend solely on the Ready item.

## Standard Item Names and Notification Control

Applications supporting OLE with direct DDE use four clipboard formats in addition to the regular data and picture formats. These are ObjectLink, OwnerLink, Native, and Binary. Binary format is a stream of bytes whose interpretation is implicit in the item; for example, the EditEnvItems, StdTargetDevice, and StdHostNames items are in Binary format. The ObjectLink, OwnerLink, and Native formats are described in Clipboard Conventions.

New items available on each topic other than the System topic are defined for this protocol. These items are the following:

Item	Description
StdDocumentName	Contains the permanent document name associated with the topic. If no permanent storage is associated with the topic, this item is empty. This item supports both request and advise transactions and can be used to detect the renaming of open documents.
EditEnvItems	Returns a list in tab-separated text format of the items that contain environmental information supported by the server for its documents. Currently defined items are StdHostNames, StdDocDimensions, and StdTargetDevice. Applications can declare other items (and define their interpretations if Binary format is used) to permit clients that are informed of these items to provide more detailed information. Servers that cannot use particular items should omit their names from the EditEnvItems item. Clients should use the WM_DDE_REQUEST message with this item to find out which items the server can use and should supply the data through a WM_DDE_POKE message.
StdHostNames	Accepts information about the client application, in Binary format interpreted as the following structure:

```
struct {  
    WORD clientNameOffset;  
    WORD documentNameOffset;  
    BYTE data[];  
};
```

```
} StdHostNames;
```

The offsets are relative to the start of the data array. They indicate the starting point for the appropriate information in the array.

StdTargetDevice

Accepts information about the target device that the client is using. This information is in Binary format, interpreted as the following structure. Offsets are relative to the start of the data array.

```
typedef struct _OLETARGETDEVICE {  
    WORD otdDeviceNameOffset;  
    WORD otdDriverNameOffset;  
    WORD otdPortNameOffset;  
    WORD otdExtDevmodeOffset;  
    WORD otdExtDevmodeSize;  
    WORD otdEnvironmentOffset;  
    WORD otdEnvironmentSize;  
    BYTE otdData[];  
} OLETARGETDEVICE;
```

StdDocDimensions

Accepts information about the size of a document. This information is in Binary format, interpreted as the following structure. These values are specified in MM\_HIMETRIC units.

```
struct {  
    int iXContainer;  
    int iYContainer;  
} StdDocDimensions;
```

StdColorScheme

Returns the colors that the server is currently using and accepts information about the colors that the client requests the server to use. This information is in Binary format, interpreted as a LOGPALETTE structure.

null

Specifies a request or advise transaction on all data contained in the topic. This item is a zero-length item name.

The update method used for advise transactions on items follows a convention in which an update specifier is appended to the actual item name. The item is encoded as follows:

itemname/update type

For backward compatibility, omitting the update type has the same result as specifying /Change. The update type placeholder may be filled with one of the following values:

Value	Meaning
/Change	Notify for each change.
/Close	Notify when document is closed.
/Save	Notify when document is saved.

DDE server applications are required to save each occurrence of a WM\_DDE\_ADVISE message that specifies a unique combination of itemname, update type, format, and conversation. A notification is disabled by a WM\_DDE\_UNADVISE message with corresponding parameters. If the WM\_DDE\_UNADVISE message does not specify a format, it disables the oldest notification in first in, first out (FIFO) rotation.

#### Standard Commands in DDE Execute Strings

The syntax for standard commands sent in execute strings is the same as for other DDE commands:

```
command(argument1,argument2,...)[command2(argument1,argument2,...)]
```

Commands without arguments do not require parentheses. String arguments must be enclosed in double quotes.

#### International Execute Commands

DDE execute strings are typically sent from a macro language in an external application and are typically localized. OLE execute commands, however, are sent by application programs for their own purposes, need not be localized, and must be commonly recognized.

The OLE standard execute commands should not be localized; the U.S. spelling and separator characters are used. Therefore, the following rules apply:

- Client applications and the client library send standard execute commands in U.S. form.
- The server library must receive the U.S. form for these commands.
- Servers written directly to the DDE-level protocol should parse the U.S. form, if they have no additional commands.
- Servers that support both OLE and localized DDE execute commands should first parse the string by using localized separators. If this fails, they should parse it again using the U.S. form and, if successful, should execute the command. Optionally, if the command is received in the U.S. form, the server can check that the command is one of the valid standard commands.

#### Required Commands

This section lists commands that must be supported by server applications.

The StdNewDocument, StdNewFromTemplate, StdEditDocument, and StdOpenDocument commands all make the document available for DDE conversations with the name DocumentName. They do not show any window associated with the document; the client must send the StdShowItem and StdDoVerbItem commands, or the StdDoVerbItem command alone to make the window visible. This enables the client to negotiate additional parameters with the server (for example, the StdTargetDevice item) without causing unnecessary repaints.

StdNewDocument(ClassName, DocumentName)

Creates a new, empty document of the given class, with the given name, but does not save it. The server should return an error value if the document name is already in use. When the client receives this error, it should generate another name and try again.

The server should not show the window until it receives a StdShowItem command. Waiting for the client to send the StdShowItem and StdDoVerbItem commands makes it possible for the client to negotiate additional parameters (for example, by using StdTargetDevice) without forcing the window to repaint.

StdNewFromTemplate(ClassName, DocumentName, TemplateName)

Creates a new document of the given class with the given document name, using the template with the given permanent name (that is, filename).

The server should not show the window until it receives a StdShowItem command. Waiting for the client to send a StdShowItem command makes it possible for the client to negotiate additional parameters (for example, by using StdTargetDevice) without forcing the window to repaint.

StdEditDocument(DocumentName)

Creates a document with the given name and prepares to accept data that is poked into it with WM\_DDE\_POKE. The server should return an error if the document name is already in use. When the client receives this error, it should generate another name and try again.

The server should not show the window until it receives a StdShowItem command. Waiting for the client to send a StdShowItem command makes it possible for the client to negotiate additional parameters (for example, by using StdTargetDevice) without forcing the window to repaint.

StdOpenDocument(DocumentName)

Sent to the System topic. This command opens an existing document with the given name.

The server should not show the window until it receives a StdShowItem command. Waiting for the client to send a StdShowItem command makes it possible for the client to negotiate additional parameters (for example, by using StdTargetDevice) without forcing the window to repaint.

StdCloseDocument(DocumentName)

Sent to the System topic. This command closes the window associated with the document. Following acknowledgment, the server terminates any conversations associated with the document. The server should not activate the window while closing it.

StdShowItem(DocumentName, ItemName [, fDoNotTakeFocus])

Sent to the System topic. This command makes the window containing the named document visible and scrolls to show the named item (if any). The optional third argument indicates whether the server should take the focus and bring itself to the front. This argument should be TRUE if the server should not take the focus; otherwise, it should be FALSE. The default value is FALSE.

StdExit

Shuts down the server application. This command should be used only by the client application that launched the server. This command is available in the System topic only.

StdExit is sent to shut down an application if an error occurs during the startup phase or if the client started the server for an invisible update. If servers have unsaved data opened by the user, they should ignore this command.

#### Variants on Required Commands

The following variants of the above commands may be sent to the document topic rather than the System topic. This allows a client that already has a conversation with the document to avoid opening an additional conversation with the system. The document name is omitted from these commands because it is implied by the conversation topic and because it may have been changed by the server. This kind of name change does not invalidate the conversation. The client should not be forced to keep track of the name change unnecessarily. However, the server must be able to use the conversation information to identify the document on which to operate.

##### StdCloseDocument

Sent to the document conversation. This command closes the document associated with the conversation without activating it. This command causes a WM\_DDE\_TERMINATE message to be posted by the server window following the acknowledgment.

##### StdDoVerbItem(ItemName, iVerb, fShow, fDoNotTakeFocus)

Sent to the document conversation. This command is similar to the StdShowItem command, except that it includes an integer indicating which of the registered operations to perform and a flag indicating whether to show the window. The server can ignore the fShow flag, if necessary.

##### StdShowItem(ItemName [, fDoNotTakeFocus])

Sent to the document conversation. This command shows the document window, scrolling if necessary to bring the item into view. If the item name is NULL, scrolling does not occur. The optional second argument indicates whether the server should take the focus and bring itself to the front. This argument should be TRUE if the server should not take the focus; otherwise, it should be FALSE. The default value is FALSE.

## The Windows Shell Overview (3.1)

This topic describes features of the shell for the Microsoft Windows operating system. The following features are supported by the dynamic-link library SHELL.DLL:

Association Functions

Drag-Drop

Icon Extraction

Registration Database



## Registration Database

The registration database is a systemwide source of information about applications. This information is used to support the integration of applications with Windows File Manager and is used by applications that support object linking and embedding (OLE).

An application can use the registration database to store the following information:

- The name of the executable file that is associated with a given filename extension
- The command line to execute--or dynamic data exchange (DDE) messages to send--when the user opens a file from Windows shell applications (File Manager or Program Manager)
- The command line to execute--or DDE messages to send--when the user prints a file from File Manager
- Details about the implementation of OLE if the application is an OLE server

The registration database is a standard part of Windows version 3.1. Any Windows version 3.0 application that supports OLE also uses the registration database. The registration database is not meant as a place for applications to store private data. Applications should use private initialization files for data that is not defined or that is not needed either by the Windows 3.1 shell applications or by OLE applications.

For most applications, the developer uses Microsoft Windows Registration Editor (REGEDIT.EXE) to edit the registration database and produce a registration (.REG) file that contains readable text strings corresponding to database entries. This .REG file can be merged into the user's registration database when the application is installed. For more information about merging text files with the database, see Format of Registration Files.

### Structure of the Database

The registration database is stored in binary format in a file named REG.DAT. This file is saved in the user's Windows directory.

Data in the registration database is in the form of a hierarchically structured tree. Each node in the tree is identified by a key name. Each key name is a string from the set of printable ASCII characters (values 32 through 127). Key names cannot include a space, a backslash (\), or a wildcard (\* or ?). Key names beginning with a period (.) are reserved.

Any key name can also be associated with a text string that provides further information about that key. The text string can contain any character from the set of printable ASCII characters. These text strings are also called values.

Each key name is unique with respect to the key that is immediately above it in the hierarchy. For example, the open and print keys are often subkeys of the key named shell. Both open and print might have subkeys named command, but open could not have two subkeys named command.

The system defines a standard entry for the root level of the database: HKEY\_CLASSES\_ROOT. Root-level key names that begin with a period are reserved by the system. Database entries that are subordinate to the HKEY\_CLASSES\_ROOT key define types (or classes) of documents and the properties that are associated with these classes. Information stored under HKEY\_CLASSES\_ROOT is used by Windows shell applications and by OLE applications.

The following table shows the structure of a typical REG.DAT file. In this table, bold characters designate reserved words and italic characters designate words or phrases that vary with the registering application.

HKEY_CLASSES_ROOT	
<i>.ext</i>	class name
<i>ClassName</i>	class description
shell	
open	
command	command line for opening application
ddeexec	DDE command used when opening document
application	DDE app name for starting conversation
topic	topic of the DDE conversation
ifexec	DDE command if initiate fails
print	
command	command line for opening application
ddeexec	DDE command used when printing document
application	DDE app name for starting conversation
topic	topic of the DDE conversation
ifexec	DDE command if initiate fails
protocol	

StdFileEditing	
server	command line for opening application
handler	path and filename for handler DLL
verb	play or edit

Future versions of the database will include more reserved words. To avoid conflict with future versions, applications should record information that is not used by the Windows shell or OLE in private initialization files.

Standardized keys help an application navigate in the database. When an application has found the key for a feature, it typically uses the text string associated with that key. (As shown in the preceding list, however, not all keys have text strings.) For example, if an application needs to display the name of an application in a dialog box, the application might use the *ClassName* key to find the *class description* text string. The class name is often an abbreviated string, for application use only, whereas the class description is the full name of the application and is presented in the user interface.

Some standard entries to the database that are occasionally used by OLE server applications are not noted in the preceding list. For more information about these standard entries, see Object Linking and Embedding Overview.

The following illustration shows how Windows Paintbrush is registered in REG.DAT (as displayed when REGEDIT.EXE is started with the /v option).

### Format of Registration Files

For most applications, the developer creates a registration (.REG) file that contains the database entries. Registration Editor (REGEDIT.EXE) can then be used to merge the .REG file into the user's REG.DAT file when the application is installed on the user's system.

The following example shows the format of a .REG file that would set up Microsoft Paintbrush with the entries shown in Figure 7.1:

```
REGEDIT
```

```
This is a comment line.
```

```
HKEY_CLASSES_ROOT\PBrush = Paintbrush Picture
HKEY_CLASSES_ROOT\.bmp = PBrush
HKEY_CLASSES_ROOT\.msp = PBrush
HKEY_CLASSES_ROOT\.pcx = PBrush
HKEY_CLASSES_ROOT\PBrush\shell\print\command = pbrush.exe /p %1
HKEY_CLASSES_ROOT\PBrush\shell\open\command = pbrush.exe %1
HKEY_CLASSES_ROOT\PBrush\protocol\StdFileEditing\verb\0 = Edit
HKEY_CLASSES_ROOT\PBrush\protocol\StdFileEditing\server = pbrush.exe
```

The first line of the file must be REGEDIT, as shown. Any subsequent lines that do not begin with HKEY\_CLASSES\_ROOT are currently treated as comments by REGEDIT.EXE. For compatibility with future versions of the database, however, a comment should not begin with a backslash (\) character or with the string HKEY. Each line to be added to the database must begin with a full key name. To create a key with an associated text string, the key name must be followed by at least one space, an equal sign (=), another space, and the string. Characters following the equal sign and single space are treated as the value of the key.

When SHELL.DLL encounters the string %1 in a command, it replaces that string with the name of the document being opened or printed.

A .REG file cannot be larger than 64K.

The setup procedure for the registering application typically merges this file with the user's REG.DAT file by running REGEDIT.EXE with the /s option. (Applications that must update the database with Windows 3.0 can use REGLOAD.EXE instead of REGEDIT.EXE to merge the files. REGLOAD.EXE is smaller than REGEDIT.EXE and does not require the common dialog box dynamic-link library COMMDDL.DLL.)

### Class Registration

Database entries that are one level below the HKEY\_CLASSES\_ROOT root-level entry are defined as classes of documents. The exception to this definition is the .ext class.

Database entries that are subordinate to the class-definition entries describe the properties of a class. The database can describe two kinds of document properties for each class of document: shell properties and protocol properties.

## Registering Filename Extensions

The .ext key name defines all files with that extension as members of a specified class. The registering application specifies the document class for an extension in the text string associated with the .ext key name.

Unlike other second-level key names, the .ext key name is not a class definition. Instead, it helps associate a class with a specific filename extension. For example, a word processor application can define a .DOC filename extension with the text string wpdoc. Then, when the word processor uses wpdoc as the class name for its documents, the .DOC extension is associated with that class.

The class name is the same name used by an OLE server application when it registers itself. For example, if a voice-annotation application named TALK.EXE registered as an OLE server, the information would look like this:

```
HKEY_CLASSES_ROOT\.tlk = Talk
HKEY_CLASSES_ROOT\Talk = Talk Voice Annotation
```

Filename extensions are recorded both in the database and in the [extensions] section of WIN.INI when the user records a filename association in the Associate dialog box. The Associate dialog box is displayed when the user chooses the Associate command from the File menu in File Manager. (Although File Manager automatically records the information in both places, SHELL.DLL does not. Applications that register filename extensions in the registration database should also record the information in WIN.INI, to provide compatibility with applications written before Windows 3.1.)

File Manager uses the filename associations recorded in WIN.INI if the information is not found in the registration database. If information is duplicated in the database and WIN.INI, File Manager uses the information in the database.

## Shell Properties

Shell properties describe how a document of a given class interacts with Windows shell applications. There are two key names for shell properties: open and print. The open properties describe how the class responds to a request from a Windows shell application to open a document. The print properties describe how the class responds to a request from Print Manager to print a document.

Both the open and print key names must have the command subkey. The value assigned to command specifies the command line used to run the application. If appropriate, this value can include command-line options.

If an application supports DDE, it can also define the ddeexec subkey for either or both of the open and print key names. The text string given with the ddeexec key name is treated as a DDE command. Defining ddeexec is particularly useful if an application already supports DDE open and print commands. Using DDE messages can add flexibility, particularly for applications that support the multiple document interface (MDI), because a DDE message string can include more than one command.

The ddeexec key has three predefined subkeys: application, topic, and ifexec.

The text string given with the application key name specifies the application name to use in establishing the DDE conversation. If the registering application does not specify an application key, the shell uses the application name specified in the command key.

The text string given with the topic key name specifies the topic name of the DDE conversation. If the application does not register a topic key, the shell uses the System topic as the default topic name.

The text string given with the ifexec key name defines the DDE command to use when initiation of the DDE conversation fails (for example, if the application is not running). When the initiation fails, the command specified by the command key is carried out and then the string specified with the ifexec key is sent. (If an application does not specify a value for the ifexec key, the command specified by the command key is executed when initiation fails and the string specified with the ddeexec key is sent again.)

## Opening Files

An application should open a file in a new instance of the associated application, even if the application supports MDI. If the user has already opened the file, applications typically give the focus to the window with the file instead of obtaining a new copy of the file.

If an MDI application does not use memory efficiently when multiple instances of the application are running, the application can open the file in the existing instance, as a new MDI window.

## Printing Files

After opening the file as described in the preceding section, the application should carry out the print command. Whenever possible, applications should display the Print dialog box to give the user the opportunity to customize the print job. If this is not possible, the file should be printed immediately. Once the file is printed or the user chooses to cancel the print job, the application should close. (If the file was opened as a new MDI window, the application typically closes the window, rather than the entire application, when the print job has finished.)

## Protocol Properties

A protocol is a convention for manipulating a document or some other collection of data. Database entries that are subordinate to the protocol key name describe the properties of a protocol. Although a class can support any number of protocols, currently only one is defined. This protocol, StdFileEditing, is used by documents that support OLE.

The StdFileEditing protocol has three subkeys: server, handler, and verb.

The text string given with the server key name is a command line that an OLE client application uses to start the server application for a linked or embedded object.

The text string given with the handler key name is the name of a dynamic-link library that acts as an object handler for OLE objects. For more information about object handlers, see [Object Linking and Embedding](#).

The verb key name has subkeys that identify the kind of action a server should take when it opens an object. These subkeys are consecutive numbers, beginning with zero. The 0 subkey corresponds to the primary verb for the objects supported by the server. For example, 0 often means Edit and 1 often means Play. For more information about verbs, see [Object Linking and Embedding](#).

For example, if an application named NewApp could not use REGEDIT.EXE to set up its protocol properties, it could set them up by using the following example:

```
HKEY hkProtocol;

if (RegCreateKey(HKEY_CLASSES_ROOT, /* root*/
    "NewAppDocument\protocol\StdFileEditing", /* protocol string */
    /
    &hkProtocol) != ERROR_SUCCESS) /* protocol key handle */
    return FALSE;

RegSetValue(hkProtocol, /* handle to protocol key */
    "server", /* name of subkey*/
    REG_SZ, /* required */
    "newapp.exe", /* command to activate server */
    10); /* text string size */

RegSetValue(hkProtocol, /* handle to protocol key */
    "handler", /* name of subkey*/
    REG_SZ, /* required */
    "nwappobj.dll", /* name of object handler */
    12); /* text string size */

RegSetValue(hkProtocol, /* handle to protocol key */
    "verb\0", /* name of subkey*/
    REG_SZ, /* required */
    "Edit", /* server should edit object */
    4); /* text string size */

RegCloseKey(hkProtocol); /* close protocol key and subkeys */
```

## Server Registration in WIN.INI

When an application creates a server protocol property and saves this key in REG.DAT, SHELL.DLL also puts this information into the WIN.INI initialization file. Some applications that use linked and embedded objects were developed before the implementation of the registration database. The information in WIN.INI allows such an application to find the command line that starts the server for an object. Server registration entries in WIN.INI are also written to the registration database whenever the user starts Windows.

The server registration entries in WIN.INI are in a section headed [embedding]. If an [embedding] section does not already exist when a registering application calls the RegCloseKey function for a key, SHELL.DLL creates it. When an application calls RegCloseKey, every class-definition key in REG.DAT that is not already in the [embedding] section is added to WIN.INI, not simply the key for which RegCloseKey was called.

The server information in WIN.INI is recorded in the following form:

```
[embedding]
ClassName=comment,textual class name,path/arguments,Picture
```

The keyword Picture indicates that the server can produce metafiles for use when rendering objects. Because commas are used as field separators, none of the fields can contain a comma.

A server can register only the name and arguments for its executable file, rather than the entire path, if the application is always installed in a directory that is mentioned in the PATH environment variable. Usually, registering the path and filename is less ambiguous than registering only the filename.

When the database is opened, the shell library reads the [embedding] section of WIN.INI and updates the registration database with any new information it contains. If the [embedding] section contains information that conflicts with REG.DAT, the information in REG.DAT is overwritten. When the database is closed, the shell library writes the information in REG.DAT back into the [embedding] section of WIN.INI. This ensures that applications that depend on WIN.INI for information about linked and embedded objects retrieve current information and that new OLE applications can simply read from and write to REG.DAT.

#### Querying and Deleting Database Entries

An application can use the RegCreateKey and RegSetValue functions to add keys to the registration database and the RegCloseKey function to indicate that a key is no longer needed by the application. Other registration functions allow an application to query the contents of the database and delete keys.

An application can use the RegEnumKey function to determine the subkeys of a specified key. Because the first parameter of RegEnumKey must be the handle of an open key, this function is typically preceded by a call to the RegOpenKey function and followed by a call to RegCloseKey. (Because the HKEY\_CLASSES\_ROOT key is always open, bracketing RegEnumKey with RegOpenKey and RegCloseKey is not strictly necessary when HKEY\_CLASSES\_ROOT is specified as the first parameter of RegEnumKey. Using RegOpenKey and RegCloseKey is a time optimization in this case, however.) The RegQueryValue function retrieves the text string that has been associated with a key name.

The following example uses the RegEnumKey function to put the values associated with top-level keys into a list box:

```
HKEY hkRoot;
char szBuff[80], szValue[80];
static DWORD dwIndex;
LONG cb;

if (RegOpenKey(HKEY_CLASSES_ROOT, NULL, &hkRoot) == ERROR_SUCCESS) {
    for (dwIndex = 0; RegEnumKey(hkRoot, dwIndex, szBuff,
        sizeof(szBuff)) == ERROR_SUCCESS; ++dwIndex) {
        if (*szBuff == '.')
            continue;
        cb = sizeof(szValue);
        if (RegQueryValue(hkRoot, (LPSTR) szBuff, szValue,
            &cb) == ERROR_SUCCESS)
            SendDlgItemMessage(hDlg, ID_ENUMLIST, LB_ADDSTRING, 0,
                (LONG) (LPSTR) szValue);
        RegCloseKey(hkRoot);
    }
}
```

The following example uses the RegQueryValue function to retrieve the name of an object handler and then calls the RegDeleteKey function to delete the key if its value is nwappobj.dll:

```
char szBuff[80];
LONG cb;
HKEY hkStdFileEditing;
```

```
if (RegOpenKey(HKEY_CLASSES_ROOT,  
"NewAppDocument\\protocol\\StdFileEditing",  
&hkStdFileEditing) == ERROR_SUCCESS) {  
  
    cb = sizeof(szBuff);  
    if (RegQueryValue(hkStdFileEditing,  
"handler",  
szBuff,  
&cb) == ERROR_SUCCESS  
&& lstrcmpi("nwappobj.dll", szBuff) == 0)  
        RegDeleteKey(hkStdFileEditing, "handler");  
        RegCloseKey(hkStdFileEditing);  
}
```

## Drag-Drop Feature

When an application implements the drag-drop feature, a user can select one or more files in File Manager, drag them to an open application, and drop them there. The application in which the files were dropped receives a message it can use to retrieve the filenames and the coordinates of the point at which the files were dropped.

The drag-drop feature depends upon SHELL.DLL. The drag-drop feature does not depend in any way on the registration database, however.

An application that can accept dropped files from File Manager calls the DragAcceptFiles function for one or more of its windows. Then, when the user releases the mouse button to drop a file or files in the window specified in the call to DragAcceptFiles, File Manager sends the application a WM\_DROPFILES message. (File Manager does not send the WM\_DROPFILES message to an application unless the application calls DragAcceptFiles.) WM\_DROPFILES contains a handle of an internal data structure the application can query to retrieve the name of the dropped file and the coordinates of the position at which the cursor was located when the file was dropped. The application can use the DragQueryFile function to retrieve the number of files that were dropped and their names. The DragQueryPoint function returns the window coordinates of the cursor when the user released the mouse button.

To free the memory allocated by the system for the WM\_DROPFILES message, an application should call the DragFinish function when it is finished.

For example, an application can call the DragAcceptFiles function when it starts and call a drag-drop function when it receives a WM\_DROPFILES message, as shown in the following example:

```
case WM_CREATE:
    DragAcceptFiles(hwnd, TRUE);
    break;

case WM_DROPFILES:
    DragFunc(hwnd, wParam);
    break;

case WM_DESTROY:
    DragAcceptFiles(hwnd, FALSE);
    break;
```

The following example uses the DragQueryPoint function to determine where to begin to write text. The first call to the DragQueryFile function determines the number of dropped files. The loop writes the name of each file, beginning at the point returned by DragQueryPoint.

```
POINT pt;
WORD cFiles, a;
char szFile[80];

DragQueryPoint((HANDLE) wParam, &pt);

cFiles = DragQueryFile((HANDLE) wParam, 0xFFFF, (LPSTR) NULL, 0);
for(a = 0; a < cFiles; pt.y += 20, a++) {
    DragQueryFile((HANDLE) wParam, a, szFile, sizeof(szFile));
    TextOut(hdc, pt.x, pt.y, szFile, strlen(szFile));
}

DragFinish((HANDLE) wParam);
```

## Association Functions

File Manager includes an Associate dialog box that makes it possible for users to associate a filename extension with a specific application. File Manager stores these associations in the registration database and the WIN.INI initialization file. If a file has a filename extension that has been associated with an application, that application starts automatically whenever a user double-clicks that file in File Manager.

Using the FindExecutable and ShellExecute functions, applications can take advantage of such associations to find and start applications or open and print files.

An application can use the FindExecutable function to retrieve the name and handle of the executable file that is associated with a specified filename. The ShellExecute function either opens or prints a specified file, depending on the value of its lpzOp parameter. To open a document file, the function relies on the association of the filename extension.



## Extracting Icons from Executable Files

An application can use the ExtractIcon function to retrieve the handle of an icon from a specified executable file, dynamic-link library, or icon file. The following example uses the DragQueryPoint function to retrieve the coordinates of the point where a file was dropped, the DragQueryFile function to retrieve the filename of a dropped file, and the ExtractIcon function to retrieve the handle of the first icon in the file, if any:

```
HDC hdc;
HANDLE hCurrentInst, hicon;
POINT pt;
char szFile[80];

hCurrentInst = (HANDLE) GetWindowWord(hwnd, GW_HINSTANCE);

DragQueryPoint((HANDLE) wParam, &pt);

DragQueryFile((HANDLE) wParam, 0, szFile, sizeof(szFile));
hicon = ExtractIcon(hCurrentInst, szFile, 0);
if (hicon == NULL)
    TextOut(hdc, pt.x, pt.y, "No icons found.", 15);
else if (hicon == (HICON) 1)
    TextOut(hdc, pt.x, pt.y,
        "File must be .EXE, .ICO, or .DLL.", 33);
else
    DrawIcon(hdc, pt.x, pt.y, hicon);
```

## Tool Helper Library Overview (3.1)

The tool helper library (TOOLHELP.DLL) makes it easier for developers who work with the Microsoft Windows 3.1 operating system to obtain system information and control system activity. This dynamic-link library was designed to streamline the creation of Windows-hosted tools, specifically Windows-hosted debugging applications. TOOLHELP.DLL is available to applications running with Windows versions 3.0 and later.

To use the elements of TOOLHELP.DLL in an application, you must include the TOOLHELP.H header file in the application source files, link the application with TOOLHELP.LIB, and ensure that TOOLHELP.DLL is in the system path.

The following topics are related to the information in this topic:

- Debugging
- Memory management
- Windows classes
- Task management
- Interrupts

### Calling Tool Helper Functions

Most of the functions in TOOLHELP.DLL use structures to return information. The first member in each of these structures is a doubleword value named `dwSize`. This value must be initialized before an application calls the function that uses the structure; otherwise, the function fails.

The `dwSize` member enables new versions of TOOLHELP.DLL to include additional features without breaking code written for structures in Windows versions earlier than 3.1.

The THSAMPLE.C sample program demonstrates how to use some of the functions in TOOLHELP.DLL.

### Accessing Internal Windows Lists

TOOLHELP.DLL includes functions that enable you to retrieve information from the internal Windows lists. These lists include the class list, module list, and task queue.

#### Walking the Windows Class List

The `ClassFirst` function fills a `CLASSENTRY` structure with information about the first class on the Windows class list. This information includes the name of the class and the instance handle of the task that owns the class.

You use `ClassFirst` to begin a walk through the Windows class list. The `ClassNext` function continues the walk by filling a `CLASSENTRY` structure with information about the next class on the Windows class list.

You use the `GetClassInfo` function to obtain more specific class information. `GetClassInfo` requires the instance handle provided by `ClassFirst` or `ClassNext` in the `CLASSENTRY` structure.

#### Walking the Windows Module List

The `ModuleFirst` function fills a `MODULEENTRY` structure with information about the first module on the list of all currently loaded modules. This information includes the module name, handle, reference count, path to the executable file, and so on.

You use `ModuleFirst` to begin a walk through the Windows module list. The `ModuleNext` function continues the walk by filling a `MODULEENTRY` structure with information about the next module on the list.

The `ModuleFindHandle` function fills a `MODULEENTRY` structure with information about a module whose handle is known. The `ModuleFindName` function fills a `MODULEENTRY` structure with information about a module whose name is known. You use `ModuleFindHandle` or `ModuleFindName`, rather than `ModuleFirst`, to begin a walk through the Windows module list at a specific module, rather than at the first module on the list.

#### Walking the Windows Task Queue

The `TaskFirst` function fills a `TASKENTRY` structure with information about the first task in the Windows task queue. This information includes the task handle, SS register value, SP register value, stack dimensions, number of pending events, PSP offset, and so on.

You use `TaskFirst` to begin a walk through the Windows task queue. The `TaskNext` function continues the walk by filling a `TASKENTRY` structure with information about the next task in the task queue.

The TaskFindHandle function fills a TASKENTRY structure with information about a task whose handle is known. You use TaskFindHandle, rather than TaskFirst, to begin a walk through the Windows task queue at a specific task, rather than at the first task in the queue.

#### Obtaining Advisory Information

To simplify system analysis, TOOLHELP.DLL includes functions that retrieve general information about the USER heap, GDI heap, memory manager, and virtual timer.

The SystemHeapInfo function fills a SYSHEAPINFO structure with information about the USER and GDI heaps. This information includes the percentage of free space and the segment handle for each heap.

The MemManInfo function fills a MEMMANINFO structure with status and performance information about the memory manager. This information includes the size of the largest free memory object, the maximum number of pages available, the maximum number of lockable pages, total linear space, total unlocked pages, number of pages in the system swap file, and so on.

The TimerCount function fills a TIMERINFO structure with the execution times of the current task and virtual machine (VM).

#### Walking the Global and Local Heaps

TOOLHELP.DLL includes functions that enable a developer to examine objects on the global and local heaps.

##### Walking the Global Heap

The GlobalInfo function fills a GLOBALINFO structure with information about the global heap. This information includes the total number of items, the number of free items, and the number of "least recently used" (LRU) items on the global heap. The information enables the application to determine how much memory to allocate for a global-heap walk. The application must allocate the memory before starting the walk. If the application allocates any memory after starting the walk, the results of the heap walk will be corrupt.

The GlobalFirst function fills a GLOBALENTY structure with information about the first object on the global heap. This information includes the structure size, the size and address of the object, the lock count, and so on.

You use GlobalFirst to begin a walk through the global heap. The GlobalNext function continues the walk by filling a GLOBALENTY structure with information about the next object on the global heap.

The GlobalEntryHandle function fills a GLOBALENTY structure with information about a global object whose handle or selector is known. The GlobalEntryModule function fills a GLOBALENTY structure with information about a specific segment in a module. You use GlobalEntryHandle or GlobalEntryModule, rather than GlobalFirst, to begin a walk through the global heap at a specific object, rather than at the first object on the global heap.

##### Walking the Local Heap

The LocalInfo function fills a LOCALINFO structure with the total number of items on the local heap. This information enables the application to determine how much memory to allocate for a local-heap walk. The application must allocate the memory before starting the walk. If the application allocates any memory after starting the walk, the results of the heap walk will be corrupt.

The LocalFirst function fills a LOCALENTY structure with information about the first object on the local heap. This information includes the structure size; the handle, address, and size of the object; the lock count; and so on.

You can use LocalFirst to begin a walk through the local heap. The LocalNext function continues the walk by filling a LOCALENTY structure with information about the next object on the local heap.

#### Tracing the Windows Stack

The StackTraceFirst function fills a STACKTRACEENTRY structure with information about the first stack frame for an inactive task. This information includes the stack-frame module handle, segment number, register contents, frame type, and so on.

You use StackTraceFirst to begin a stack trace of an inactive task. The StackTraceNext function continues the stack trace by filling a STACKTRACEENTRY structure with information about the task's next stack frame.

The StackTraceCSIPFirst function fills a STACKTRACEENTRY structure with information about a stack frame whose SS:BP and CS:IP values are known. You should use StackTraceCSIPFirst, rather than StackTraceFirst, to begin a stack trace of an active task.

#### Examining and Modifying Memory Contents

TOOLHELP.DLL includes functions that enable you to examine and modify global memory contents without consideration for selector tiling and aliasing or read-write attributes.

The MemoryRead function reads global memory at a specific selector and offset. The MemoryWrite function writes to global memory at a specific selector and offset.

The GlobalHandleToSel function converts a global memory handle to a selector.

#### Installing Callback Functions

TOOLHELP.DLL includes functions that enable you to trap an application's interrupts and notifications.

The InterruptRegister function installs a callback function that handles all system interrupts. The callback function must be reentrant and must explicitly preserve all register values. The InterruptUnRegister function restores the default processing.

The NotifyRegister function installs a notification callback function for a specific task. Typically, the notification callback function cannot use any Windows functions except the TOOLHELP.DLL functions and the PostMessage function. The NotifyUnRegister function restores the default processing.

The exit code returned by a non-Windows application may reflect an error encountered by Windows when it attempted to start the application, rather than a value returned by the application itself. These error values are as follows:

Error value	Cause
0x81	Could not start the application because of a file-access problem. This problem originated either in the application or its PIF file. Following are likely reasons for this error value:  File not found Path not found No file handles Invalid drive Access denied Sharing violation Invalid executable format
0x82	Could not start the application, because of insufficient memory or disk space.
0x83	Abnormal termination.
0x84	Could not start the application, because of incorrect version.
0x85	Could not start the application, because MS-DOS Interrupt 21h Function 4B00h (Load and Execute Program) failed.
0x86	Could not start the application, because the TOOLHELP.DLL task-switching functions prevented it from starting.

#### Controlling Process Execution

TOOLHELP.DLL includes four functions you can use to control process execution: TaskGetCSIP, TaskSetCSIP, TaskSwitch, and TerminateApp. These functions are designed for use exclusively in Windows-hosted debuggers.

When an inactive task is activated, it begins execution at the location specified by its CS:IP value. The TaskSetCSIP function sets this value, and the TaskGetCSIP function returns the value.

The TaskSwitch function activates a specific task beginning at a specified CS:IP value.

The TerminateApp function terminates an application as if a general protection (GP) fault had occurred.

## LZExpand Overview (3.1)

The Microsoft Windows operating system includes the dynamic-link library LZEXPAND.DLL. Typically, an application calls functions in LZEXPAND.DLL to decompress data previously compressed by Microsoft File Compression Utility (COMPRESS.EXE).

A version of LZEXPAND.DLL was shipped with Windows version 3.0. That version of LZEXPAND.DLL does not contain the full set of functions that is included with the Windows 3.1 version. Applications that could be installed on a system running Windows 3.0 should always check the version number of the library to ensure that the correct version is being used. For more information about checking version numbers, see the [File Installation Library](#).

This topic describes important concepts relating to data compression and describes the decompression functions in LZEXPAND.DLL.

### Data Compression

Data compression is an operation that reduces the size of a file by minimizing redundant data. In a file that contains text, redundant data could be frequently occurring characters, such as the space character, or common vowels, such as the letters e and a; it could also be frequently occurring character strings. Data compression operations create a compressed version of a file by minimizing this redundant data.

Each of the many types of data-compression operations minimizes redundant data in a unique manner. For example, the Huffman encoding algorithm assigns a code to characters in a file based on how frequently those characters occur. Another compression algorithm, called run-length encoding, generates a two-part value for repeated characters: The first part specifies the number of times the character is repeated, and the second part identifies the character. Another compression algorithm, known as the Lempel-Ziv algorithm, converts variable-length strings into fixed-length codes, which consume less space than the original strings.

To compress large applications or data files, you can run COMPRESS.EXE from the Microsoft MS-DOS@command line. COMPRESS.EXE uses the Lempel-Ziv compression algorithm.

### Data Decompression

Applications can call the functions in LZEXPAND.DLL to decompress files compressed with COMPRESS.EXE. The functions can also process uncompressed files without attempting to decompress them. To use these functions, include the LZEXPAND.H header file. To use the static-link libraries, define LIB before including LZEXPAND.H.

For a list of functions contained in LZEXPAND.DLL, see the [Lempel-Ziv Encoding Functions](#) topic.

### Decompressing a Single File

An application can decompress a single compressed file by performing the following tasks:

- 1 Open the compressed file by calling the [LZOpenFile](#) function or a combination of the [OpenFile](#) and [LZInit](#) functions.
- 2 Open the destination file by calling the [LZOpenFile](#) or [OpenFile](#) function.
- 3 Copy the source file to the destination file by calling the [LZCopy](#) function and passing the handles returned by [LZOpenFile](#) (or [LZInit](#)).
- 4 Close the files by calling the [LZClose](#) function.

### Decompressing Multiple Files

An application can decompress multiple files by performing the following tasks:

- 1 Open the source file by calling the [LZOpenFile](#) function or a combination of the [OpenFile](#) and [LZInit](#) functions.
- 2 Open the destination file by calling the [LZOpenFile](#) or [OpenFile](#) function.
- 3 Allocate memory for the copy operation by calling the [LZStart](#) function.
- 4 Copy the source files to the destination files by calling the [CopyLZFile](#) function.
- 5 Release the allocated memory by calling the [LZDone](#) function.
- 6 Close the files by calling the [LZClose](#) function.

### Reading Bytes from Compressed Files

In addition to decompressing a complete file at a time, an application can decompress compressed files a portion at a time by using the [LZSeek](#) and [LZRead](#) functions. These functions are particularly useful

when it is necessary to extract parts of large files. For example, a font manufacturer may have compressed files containing font metrics in addition to character data. To use the information in these files, an application would need to decompress the file; however, most applications would use only part of the file at any particular time. When the user queried the font metrics, the application would extract data from the header. When the user rendered text output, the application would reposition the file pointer by calling LZSeek and extract the character data.

## Stress Library (3.1)

The system resources stress-testing library (STRESS.DLL) is a dynamic-link library that artificially consumes system resources, enabling developers to observe how an application behaves in scarce-resource conditions. This library was designed to make scarce-resource testing easier and more realistic. It is used by the STRESS.EXE utility.

### System Resources Stress-Testing Library Functions

Following are the system resources affected by STRESS.DLL, with the functions that consume and release each resource:

Resource	Allocation function	Release function
Global memory	<u>AllocMem</u>	<u>FreeAllMem</u>
GDI heap memory	<u>AllocGDIMem</u>	<u>FreeAllGDIMem</u>
User heap memory	<u>AllocUserMem</u>	<u>FreeAllUserMem</u>
Disk space	<u>AllocDiskSpace</u>	<u>UnAllocDiskSpace</u>
File handles	<u>AllocFileHandles</u>	<u>UnAllocFileHandles</u>

## File Installation Overview (3.1)

The file installation library in the Microsoft Windows version 3.1 operating system makes it easier for applications to install files properly and enables utility programs to analyze files that are currently installed.

This overview is divided into three parts:

File Installation Concepts

Creating an Installation Program

Adding Version Information to a File



## File Installation Concepts

The file installation library includes functions that determine where a file should be installed, identify conflicts with currently installed files, and perform the installation process. These functions enable installation programs to avoid the following problems:

- Installing older versions of components over newer versions
- Changing the language in a mixed-language system without notification
- Installing multiple copies of a library in different directories
- Copying files to network directories shared by multiple users

The file installation library also includes functions that enable applications to query a version resource for information about a file and present the information to the user in a clear format. This information includes the file's purpose, author, version number, and so on. (For more information about version resources, see [Adding Version Information to a File](#)).

The file installation library is available for Windows and non-Windows applications. Windows applications should use the dynamic-link library VER.DLL and the header file VER.H. Non-Windows applications should use one of the following static-link libraries: VERS.LIB, VERC.LIB, VERM.LIB, or VERL.LIB. Applications that use the static-link libraries should use the following line before including VER.H:

```
#define LIB
```

## Creating an Installation Program

An installation program typically has the following goals:

- To place files in the correct location
- To notify the user if the installation program is replacing an existing file with a version that is significantly different--for example, replacing a German file with an English file, or replacing a newer file with an older file

When writing the installation program, you must have the following information for each file on the installation disk(s):

- The name and location of the file (referred to as the source file).
- The name of the equivalent file on the user's hard disk (referred to as the destination file). This name is usually the same as the filename on the installation disk.
- The sharing status of the file--that is, whether the file is private to the application being installed or could be shared by multiple applications.

For each file on the installation disk(s), the installation program must, at least, call the VerFindFile and VerInstallFile functions. These functions are described briefly in the rest of this section.

You use the VerFindFile function with the destination-file name to determine where the file should be copied to on the disk. This function also enables you to specify whether the file is private to the application or can be shared. If a problem occurs in finding the file, VerFindFile returns an error value. For example, if Windows is using the destination file, VerFindFile returns `VFF_FILEINUSE`. The installation program must notify the user of the problem and respond to the user's decision to continue or end the installation.

The VerInstallFile function copies the source file to a temporary file in the directory specified by VerFindFile. If necessary, VerInstallFile expands the file by using the functions in the data decompression library, `LZEXPAND.DLL`.

VerInstallFile compares the version information of the temporary file to that of the destination file. If they differ, VerInstallFile returns one or more error values. For example, it returns `VIF_SRCOLD` if the temporary file is older than the destination file and `VIF_DIFFLANG` if the files have different language identifiers or code-page values. The installation program must notify the user of the problem and respond to the user's decision to continue or end the installation.

Some VerInstallFile errors are recoverable. That is, the installation program can call VerInstallFile again, specifying the `VIFF_FORCEINSTALL` option, to install the file regardless of the version conflict. If VerInstallFile returns `VIF_TEMPFILE` and the user chooses not to force the installation, the installation program should delete the temporary file.

VerInstallFile could encounter a nonrecoverable error when attempting to force installation, even though the error did not exist previously. For example, the file could be locked by another user before the installation program tried to force installation. If an installation program attempts to force installation after a nonrecoverable error, VerInstallFile fails. The installation program must deal with this situation.

The recommended solution is to display a common dialog box with the buttons `Install`, `Skip`, and `Install All` for all errors. The `Install All` button should prevent the installation program from prompting the user about similar errors by including the `VIFF_FORCEINSTALL` option in all subsequent uses of VerInstallFile. For nonrecoverable errors, the `Install` and `Install All` buttons should be disabled.

To display a useful error message to the user, the installation program usually must retrieve information from the version resources of the conflicting files. The file installation library provides four functions the installation program can use for this purpose: GetFileVersionInfoSize, GetFileVersionInfo, VerQueryValue, and VerLanguageName. The GetFileVersionInfoSize function returns the size of the version information. The GetFileVersionInfo function then uses information retrieved by GetFileVersionInfoSize to retrieve a structure that contains the information. The VerQueryValue function retrieves a specific member from that structure.

For example, if VerInstallFile returns the `VIF_DIFFTYPE` error, the installation program should use GetFileVersionInfoSize, GetFileVersionInfo, and VerQueryValue on the temporary and destination files to obtain the general type of each file. If the languages of the files conflict, the installation program should also use the VerLanguageName function to translate the binary language identifier into a text representation of the language. (For example, `0x040C` translates to the string `French`.)

If VerInstallFile returns a file error, such as `VIF_ACCESSVIOLATION`, the installation program should use `MS-DOS Interrupt 21h Function 59h (Get Extended Error)` to obtain the most recent error value. The program should translate this value into an informative message to display to the user. The program must not yield control between calling VerInstallFile and calling `Get Extended Error`. If it does, the MS-DOS

error value could reflect a later error. (An error could also occur while the program is making the MS-DOS call.)

## Adding Version Information to a File

Version information can be added to any Windows file that can have Windows resources, such as a dynamic-link library, an executable file, or a font file. To add the information, you must create a version resource and add the resource to the file by using RC.

## 32-Bit Memory Management Library

One of the significant features of 80386 and 80486 processors is the availability of 32-bit registers for the manipulation of code and data. Applications written to use these registers can avoid the segmented memory model of earlier CPUs and instead use a flat memory model in which memory is viewed as a single, contiguous block.

Although the Microsoft Windows operating system continues to adhere to a segmented 16-bit memory model, Windows does provide a set of functions that allow an application to make use of the 32-bit memory-addressing capabilities of the 80386 and 80486 processors. These functions are available to an application through a dynamic-link library (DLL) named WINMEM32.DLL.

Your application's installation program should use the file installation library (VER.DLL) to ensure that it does not install an older version of WINMEM32.DLL over a newer version. For more information about VER.DLL, see File Installation Overview.

This topic introduces the functions contained in WINMEM32.DLL and explains how to use these functions in the context of a Windows application. It covers the following information:

- Some of the differences between a segmented memory model and a flat memory model
- Use of WINMEM32.DLL to take advantage of the 32-bit memory-addressing capabilities of 80386 and 80486 processors
- Programming considerations for use of 32-bit memory in a Windows application
- Use of 32-bit memory in a Windows application
- A directory of WINMEM32.DLL functions
- Assembly-language examples illustrating how to use WINMEM32.DLL functions

You should be thoroughly familiar with the following information about 80386 and 80486 processors that is not covered in this topic:

- Terminology and concepts relating to the architecture
- Code-management features
- Memory-management features

Only developers with experience writing Windows applications and assembly language code should attempt to use these functions in an application.

### Segmented and Flat Memory Models

The family of processors that includes 80286, 80386, and 80486 processors implements a segmented memory model in which system memory is divided into 64K segments. In the real mode of these processors, the address of any byte consists of two 16-bit values: a segment address and an offset. (Windows version 3.1 does not support real mode.) In the protected mode of the 80286, 80386, and 80486 processors, the segment address is replaced by a selector value that the processor uses to access the 64K segment. In either mode, a memory object larger than 64K occupies all or part of several segments. An application cannot access such an object as though it consisted of a single contiguous block simply by incrementing a pointer to the memory. Instead, the application can increment only the offset portion of the address, taking care not to exceed the 64K boundary of the segment.

The 80386 processor introduced 32-bit registers that parallel the 16-bit registers of older processors. These registers make it possible for the first time to access memory in segments larger than 64K. In fact, the maximum segment size is potentially so large ( $2^{32}$  bytes) that a flat memory model utilizing a single segment is now feasible. In this model, an application's code, data, or both occupy a single segment. The application can manipulate the 32-bit offset portion of the memory as though it were a simple pointer. The application can increment and decrement the offset portion of the memory throughout the address space without having to deal with multiple segment boundaries.

To a certain extent, the flat memory model most closely resembles the tiny memory model, in which both code and data occupy a single segment; of course, the segment is much larger than the 64K limit imposed by the segmented memory model. As in the tiny memory model, the beginning of the segment of the flat memory model can appear anywhere in memory. In other words, the segment-descriptor portion of the address can refer to virtually any location in memory. As the application moves through memory, the segment descriptor never changes. Only the offset is incremented and decremented to point to different locations in memory.

The flat memory model makes it possible for you to ignore segments and segment registers. The segment registers are loaded at the start of the 32-bit code and are then left alone. The rest of the application runs in this purely 32-bit offset mode--all pointers are near pointers.

It is not possible to implement a Windows application by using an exclusively flat memory model. Because Windows itself relies on the 16-bit segmented memory model, any application that interacts with Windows must implement at least one 16-bit code segment. Despite this limitation, it is possible

for a Windows application to reside largely in one or more 32-bit code segments and to use 32-bit data segments. The WINMEM32.DLL library makes this possible in a way that ensures the application cooperates fully with Windows and similar platforms. For more information, see Flat Memory Model Limitations.

#### Using the WINMEM32.DLL Library

Although you could directly implement code for a flat memory model in your Windows application, this implementation would necessarily be unique to your application. As a result, your application might not run with future versions of Windows or with other compatible platforms.

WINMEM32.DLL supplies a standard method for implementing a flat memory model that is guaranteed to run with future versions of Windows and other compatible platforms. It gives your application access to services for allocating, reallocating, and freeing 32-bit memory objects; for translating 32-bit pointers to 16-bit pointers that can be used by Windows and MS-DOS functions; and for aliasing a data segment to a code segment so you can execute code loaded into a 32-bit segment.

Your application can load WINMEM32.DLL when Windows is running in standard or 386 enhanced mode. However, because the 32-bit registers of the 80386 or 80486 processor are available only when Windows is in 386 enhanced mode, WINMEM32.DLL is enabled only in that mode. If your application runs in standard mode, you must design your application so that it can access 16-bit memory instead of 32-bit memory. You can find out which mode Windows is running in by calling the GetWinFlags function.

WINMEM32.DLL contains eight functions that enable your application to access 32-bit memory. The following table summarizes each of these functions:

Function	Description
<u>GetWinMem32Version</u>	Returns the version number of the WINMEM32.DLL application programming interface (API).
<u>Global16PointerAlloc</u>	Converts a 32-bit pointer to a 16-bit pointer.
<u>Global16PointerFree</u>	Frees a pointer alias created by the <u>Global16PointerAlloc</u> function.
<u>Global32Alloc</u>	Allocates a 32-bit memory object.
<u>Global32CodeAlias</u>	Creates a code-segment alias for a 32-bit memory object, allowing code in the object to be executed.
<u>Global32CodeAliasFree</u>	Frees a code-segment alias created by the <u>Global32CodeAlias</u> function.
<u>Global32Free</u>	Frees a 32-bit memory object.
<u>Global32Realloc</u>	Changes the size of a 32-bit memory object.

A directory listing of these functions appears later in this topic.

Because WINMEM32.DLL is a standard Windows DLL, your application loads it as it would any other DLL. Your application should be linked so that the case of the DLL entry point names is ignored.

The WINMEM32.DLL functions use the same calling conventions as other Windows functions. The DLL entry points are external FAR PASCAL procedures. They preserve the SS, BP, DS, SI, and DI registers, and they return values in AX register or DX:AX register pair.

#### Considerations for Using 32-Bit Memory

As previously noted, Windows adheres to the segmented memory model. That is, all far pointers are in the form 16:16 consisting of a 16-bit segment selector, combined with a 16-bit offset within the segment. An application using the 32-bit registers of the 80386 or 80486 processor cannot directly call the Windows functions, because its far pointers are in the form 16:32 and Windows cannot work with the extra 16 bits in the offset portion of the address.

Because of this conflict, a Windows application cannot reside exclusively in a 32-bit segment. It must contain at least one 16-bit helper code segment through which it interacts with Windows (including WINMEM32.DLL). In other words, all calls to Windows functions must be made in the helper code segment. The helper segment contains the code that converts the 16:32 pointers in the 32-bit segment to the 16:16 pointers used by Windows functions. This segment also performs the same tasks for the application when the application makes calls to MS-DOS, to other DLLs, or to any other code that uses 16:16 pointers exclusively.

An important limitation on this helper segment is that it must not be discardable (although it can be movable). If the segment is discarded and a 32-bit segment attempts to access the segment, an indirect call into the Windows kernel module to reload the segment results. Because the source of this indirect call is not a 16-bit segment, the system might crash.

Another important consideration is that in writing your application you must not assume anything about the state of the 32-bit registers around 16:16 function calls. For instance, the Windows function calls preserve SI and DI registers, but they presently do not preserve ESI and EDI registers. If the application needs to preserve 32-bit registers around 16:16 function calls, it must explicitly push and pop the register values around the calls. If the 32-bit code segment that calls a Windows function (by means of the helper segment) needs ESI and EDI registers to be preserved when the Windows function returns, the helper segment must explicitly save the registers before making the actual Windows function call. The helper segment must then restore the registers when the Windows function returns.

This rule also applies to return values when a 32-bit segment indirectly calls a Windows function and the caller requires a 32-bit return value. The helper segment must explicitly set the high-order 16 bits of the return value when it moves it into the EAX register, as shown in the following examples:

```
movzx    eax,ax ; unsigned return
```

```
movsx    eax,ax ; signed return
```

All these considerations apply equally to calls to Windows DLLs, MS-DOS, and other 16-bit functions.

### Flat Memory Model Limitations

In the Windows environment, system memory is a shared resource that Windows manages on behalf of all applications. For this reason, a true flat memory model is not possible in the Windows environment. When an application allocates 32-bit memory in Windows, the memory that Windows gives the application can be located anywhere in physical memory. The memory to which the selector refers is specific to the application and does not include systemwide memory locations. In other words, the selector that the application receives does not refer to linear address 0. This means that offset 400h for the selector does not point to the MS-DOS ROM BIOS data area, for example.

Windows applications do not need to address these systemwide memory locations directly, so there is no need to map these locations in the 32-bit memory objects.

### The Application Stack

Windows cannot operate in an environment of mixed segment types (including both 16:16 and 16:32 segments). As a result, the stack selector size must match the corresponding code selector size. When the processor is executing code in a 16:32 (USE32) code segment, the selector in the SS register must contain a 16:32 selector. When the processor is executing code in a 16:16 (USE16) segment, the SS register must contain a 16:16 selector.

When the 80386 or 80486 processor is executing on a USE16 stack segment, it uses the low-order 16 bits of the ESP register as the SP register. Because only the low-order 16 bits are of use when the processor is running on a USE16 stack segment, the processor does not control how the high-order 16 bits of the ESP register are set. As a result, the high-order 16 bits are set at random. When an application switches to a USE32 stack segment, the ESP register contains a corrupted pointer unless the high-order 16 bits of ESP are set properly.

Suppose that a Windows application has a USE32 code segment and a USE16 helper segment, but (improperly) only a USE32 stack. When the application calls from its USE32 code into the USE16 segment, the application continues to use its USE32 stack. The USE16 code segment calls a Windows function, which changes the selector in the SS register to a USE16 selector. Because the stack is now USE16, the high-order 16 bits of the ESP register are set at random. The code that originally switched stacks then restores the original selector in SS and, lacking the information that the selector referred to a USE32 stack, restores the 16-bit SP register instead of the full 32 bits of the ESP register. As a result, the USE32 stack now has an invalid pointer in the ESP register.

There are a number of ways to deal with this problem. One solution is for an application to maintain two separate stacks, one USE16 and the other USE32. Maintaining separate stacks requires you to include extra code --for example, you must copy parameters for stack-calling conventions such as that used in C. Another solution is to maintain one stack but two stack selectors, one USE16 and the other USE32, both of which point to the same memory. This requires the USE32 stack to be restricted to ESP values less than or equal to FFFFh.

In either case, the USE16 code segment must switch to the USE32 stack immediately before calling into a USE32 code segment. When control returns from the USE32 code segment to the USE16 code segment, the USE16 segment must switch back to the USE16 stack before doing anything else.

Because the problem with stack switching is the corruption of the high 16 bits of ESP, a Windows application with 16:32 code must make sure that it sets the high 16 bits of ESP when it is switching to

the USE32 stack selector. It sets these bits by placing the selector into the SS register, as shown in the following example:

```
mov    ss,word ptr [Use32StackSel]
mov    esp,dword ptr [Use32StackOffset]

mov    ss,word ptr [Use32StackSel]
movzx  esp,word ptr [Use32StackOffset]

mov    ss,word ptr [Use32StackSel]
movzx  esp,sp
```

#### Interrupt-Time Code

A 32-bit code segment in a Windows application must not contain code that is executed at interrupt time. Also, it must not contain data that is accessed at interrupt time. Any code executed at interrupt time must be in a USE16 code segment. The code must use a USE16 stack. Data used at interrupt time must be USE16 data. This rule also applies to processor exceptions (such as the coprocessor exception) because they are handled as interrupts are handled. Note, however, that it is acceptable for a 32-bit code segment to access data in a USE16 data segment.

#### Programming Languages

The helper segment has to perform very low-level tasks to manage transitions between USE16 and USE32 stacks and between USE16 and USE32 code. For this reason, it is difficult to use a high-level language such as C to write the helper segment code. Even if you write the helper segment in C, you must add assembly-language support for the more difficult tasks. In most cases, it is easier and more efficient to write the entire helper segment in assembly language.

#### Using 32-Bit Memory in a Windows Application

There are three common uses for 32-bit memory in a Windows application. In increasing order of complexity, they are:

- Using 32-bit data objects in 16-bit code
- Using 32-bit code and data in a subroutine library
- Using 32-bit code and data for the main program

The remaining topics in this section briefly describe these uses.

#### Using 32-Bit Data Objects

The simplest use of 32-bit memory is to store data that is used exclusively by USE16 code segments. In this case, the application does not require a dedicated helper segment because it contains no USE32 code segments. Instead, each of its code segments performs the necessary tasks of allocating, reallocating, and freeing the 32-bit memory. If data from the 32-bit memory is to be passed to Windows functions or other 16-bit functions, the application calls the `Global16PointerAlloc` function so that the application's USE16 code segment can perform the aliasing of 32-bit pointers to the 16-bit pointers.

#### Using 32-Bit Code and Data in a Subroutine Library

Using 32-bit segments for code and data can simplify porting an application from a 32-bit platform to the Windows environment when portions of the application can be isolated as a subroutine library. This subroutine library serves as a low-level engine but does not call Windows or MS-DOS functions.

As when the 32-bit memory is used exclusively for data storage, the USE16 code segment retains control of the program. Typically, the USE16 segment allocates the 32-bit memory, creating one or more objects for code and data. In addition to the data-management tasks described in Considerations for Using 32-Bit Memory, the USE16 segment also loads the subroutine code into one of the 32-bit segments, fixes up the pointers in the code as required, and creates a code-segment alias to permit the code to be executed. The USE16 code segment is responsible for maintaining control of the program flow, calling into the USE32 code segment when it requires the low-level services of the subroutine library.

#### Using 32-Bit Code and Data for the Main Program

The most complex use of 32-bit memory involves placing the primary control of the program in a 32-bit code segment. In this type of application, the USE16 segment is reduced to helper status exclusively. During initialization, the USE16 segment allocates the 32-bit memory for code and data, loads the code into the USE32 segment, creates a code-segment alias for the USE32 segment, and then calls the main entry point in the USE32 segment.



From then on, the USE32 segment takes control of the program, calling into the USE16 helper segment only when the application needs to call Windows or MS-DOS functions. The USE32 segment continues to control the flow of the program until the application is ready to close. Only then does it return control to the USE16 segment so the USE16 segment can free the 32-bit memory and perform other cleanup tasks before the application quits.

#### Error Values

This section describes error values returned by the functions that applications can use for 32-bit memory management. Most of these functions return zero to indicate success. The following table describes each error value:

Value	Meaning
WM32_Insufficient_Mem	Insufficient memory. There is not enough memory to satisfy the requested allocation or reallocation.
WM32_Insufficient_Sels	Selector not available. There is not enough room in the descriptor table(s) to allocate the required selector(s). It may be necessary to advise the user to close other Windows applications.
WM32_Invalid_Arg	Invalid parameter. One of the parameters was invalid. For example, a size parameter might be out of range.
WM32_Invalid_Flags	Invalid flag. The wFlags parameter contained at least one invalid bit setting. The wFlags parameter currently is not used and must be set to zero.
WM32_Invalid_Func	Invalid function. The current Windows mode does not support this function. Windows supports the 32-bit memory functions only in 386 enhanced mode.

WM32\_Insufficient\_Mem

Insufficient memory. There is not enough memory to satisfy the requested allocation or reallocation.

WM32\_Insufficient\_Sels

Selector not available. There is not enough room in the descriptor table(s) to allocate the required selector (s). It may be necessary to advise the user to close other Windows applications.

WM32\_Invalid\_Arg

Invalid parameter. One of the parameters was invalid. For example, a size parameter might be out of range.

### WM32\_Invalid\_Flags

Invalid flag. The wFlags parameter contained at least one invalid bit setting. The wFlags parameter currently is not used and must be set to zero.

WM32\_Invalid\_Func

Invalid function. The current Windows mode does not support this function. Windows supports the 32-bit memory functions only in 386 enhanced mode.

## Screen Saver Library

The Microsoft Windows operating system provides special applications called screen savers that start when the mouse and keyboard have been idle for a period of time. Screen savers exist for two main reasons:

- To avoid phosphor burn caused by static images on a screen
- To conceal sensitive information left on a screen

Clearing a screen addresses both goals, but screen savers are not restricted to this simple use. They can also display animated sequences such as a fish tank or fireworks. Animated sequences avoid phosphor burn by continually changing the image.

Windows provides a screen saver application that monitors the mouse and keyboard and starts the screen saver after a period of inactivity. The Desktop section of Windows Control Panel makes it possible for users to select from a series of screen savers, specify how much time should elapse before the screen saver is started, configure screen savers, and preview screen savers.

This topic describes how to create a custom screen saver and add it to the library of screen savers users can select by using Control Panel.

### About Screen Savers

Screen savers are Windows applications that contain specific variable declarations, exported functions, and resource definitions. The static-link library SCRNSAVE.LIB contains the WinMain function and other startup code required for a screen saver. To create a screen saver, you create a source module containing specific function and variable definitions and link it with SCRNSAVE.LIB. Your screen saver module is responsible only for configuring itself and for providing visual effects.

Screen savers are loaded automatically when Windows starts or when a user activates the screen saver feature by using Control Panel. Windows monitors keystrokes and mouse movements and starts the screen saver after a period of inactivity specified by the user.

Windows does not start the screen saver if any of the following conditions exists:

- The active application is not a Windows application.
- A computer-based training (CBT) window is present.
- The active application returns a nonzero value in response to the WM\_SYSCOMMAND message sent with the SC\_SCREENSAVE identifier.

When your screen saver starts, the startup code in SCRNSAVE.LIB creates a full-screen window. The window class for the screen saver window is declared as follows:

```
WNDCLASS cls;  
  
cls.hCursor = NULL;  
cls.hIcon = LoadIcon(hInst, MAKEINTATOM(ID_APP));  
cls.lpszMenuName = NULL;  
cls.lpszClassName = "WindowsScreenSaverClass";  
cls.hbrBackground = GetStockObject(BLACK_BRUSH);  
cls.hInstance = hInst;  
cls.style = CS_VREDRAW | CS_HREDRAW  
| CS_SAVEBITS | CS_DBLCLKS;  
cls.lpfnWndProc = ScreenSaverProc;  
cls.cbWndExtra = 0;  
cls.cbClsExtra = 0;
```

Your source module provides the ScreenSaverProc window procedure. Your resource-definition file supplies the icon identified by ID\_APP. This icon is visible only when your screen saver is run as a stand-alone application. (To be run by Control Panel, a screen saver must have the .SCR filename extension; to be run as a stand-alone application, it must have the .EXE filename extension.)

### Creating a Screen Saver

The SCRNSAVE.H header file defines the function prototypes for the screen saver functions in SCRNSAVE.LIB. You must include this header file in your source module.

You must also define the idsAppName string. The idsAppName string should contain a screen saver name of the form Screen Saver.Name, where Name is a unique name for your screen saver. For example, a screen saver named Bouncer would include the following line in the STRINGTABLE statement in its resource-definition file:

```

STRINGTABLE PRELOAD
BEGIN
    idsAppName"Screen Saver.Bouncer"
    .
    . /* other strings */
    .
END

```

If your screen saver stores configuration information, it should use the `idsAppName` string as the application heading for the configuration block in the CONTROL.INI file. For more information about storing screen saver configuration information, see [Providing a Configuration Routine](#).

Your application should declare the following global variables, which are defined in SCRNSAVE.LIB:

```

extern HINSTANCE hMainInstance;
extern HWND hMainWindow;

```

The `hMainInstance` variable contains the instance handle of your application. The `hMainWindow` variable contains the window handle of the screen saver window.

#### Processing Screen Saver Messages

Your screen saver module must include a `ScreenSaverProc` window procedure to receive and process messages for the screen saver window. The `ScreenSaverProc` window procedure must pass unprocessed messages to the `DefScreenSaverProc` function rather than to the `DefWindowProc` function.

Your `ScreenSaverProc` window procedure can substitute its own actions for the message handling performed by `DefScreenSaverProc`.

The `ScreenSaverProc` window procedure must be exported by including it in the EXPORTS section of your `module-definition (.DEF)` file.

#### Providing a Configuration Routine

When the user chooses the Setup button, Windows uses the `/c` or `-c` command-line option to start the screen saver. To start the screen saver without displaying the configuration dialog box, Windows uses the `/s` or `-s` command-line option. When no command-line option is used, Windows displays the configuration dialog box, just as if `/c` had been specified.

If your screen saver supports configuration by the user, your source module must provide the following functions and dialog box resource to handle configuration:

Name	Description
<code>ScreenSaverConfigureDialog</code>	Dialog box procedure for a configuration dialog box.
<code>RegisterDialogClasses</code>	Function that registers any special or nonstandard window classes needed for a configuration dialog box.
<code>DLG_SCRNSAVECONFIGURE</code>	Dialog box template for a configuration dialog box.

When Windows starts your screen saver with the configuration option (`/c`), the `WinMain` function in SCRNSAVE.LIB calls the `RegisterDialogClasses` function and then displays the configuration dialog box.

Define the `ScreenSaverConfigureDialog` function as you would any dialog box procedure.

Your screen saver should save its configuration settings in the CONTROL.INI file. SCRNSAVE.LIB uses the application name stored in the `idsAppName` `STRINGTABLE` statement as the CONTROL.INI application heading. Your application can use the `LoadString` function to retrieve the name of the heading from CONTROL.INI and then use the `WritePrivateProfileString` and `WritePrivateProfileInt` functions to store the configuration information.

The `hInst` parameter of the `RegisterDialogClasses` function contains the instance handle for the screen saver. This is the same value contained in the `hMainInstance` global variable. If your configuration routine does not require any special window classes, your `RegisterDialogClasses` function can simply return TRUE.

#### Creating Module-Definition and Resource-Definition Files

Be sure to export the `ScreenSaverProc` function and, if it is present, the `ScreenSaverConfigureDialog` function. The `RegisterDialogClasses` function should not be exported.

The DESCRIPTION statement in your module-definition file must use the following format:

```
DESCRIPTION 'SCRNSAVE : description'
```



If your screen saver includes a configuration routine, you should include a dialog box template with the DLG\_SCRNSAVECONFIGURE identifier.

#### Installing New Screen Savers

Control Panel searches the Windows startup directory for files with the .SCR extension when compiling the list of available screen savers. (Screen saver applications are standard Windows executable files. Simply rename the compiled screen saver so that its extension is .SCR.)

#### A Sample Screen Saver

The remainder of this topic discusses the implementation of a screen saver application.

#### General-Purpose Declarations

Screen savers must use the string identifier idsAppName to identify themselves for other routines in SCRNSAVE.LIB:

```
STRINGTABLE PRELOAD
BEGIN
    idsAppName "Screen Saver.ScreenSaverName"
    .
    /* other strings */
    .
END
```

The idsAppName string contains the name of the screen saver. The name to the right of the period is a unique name for the screen saver. The screen saver application can retrieve this string by calling the LoadString function.

Screen savers must also declare the following external variables:

```
HINSTANCE hMainInstance;
HWND hMainWindow;
```

These external variables are defined in SCRNSAVE.LIB. They contain handles to the application instance and main window.

#### Message Handling

The following ScreenSaverProc function processes the WM\_CREATE, WM\_TIMER, WM\_DESTROY, and WM\_ERASEBKGD messages before calling the DefScreenSaverProc function:

```
LONG CALLBACK ScreenSaverProc(HWND hwnd, UINT msg, WPARAM wParam,
    LPARAM lParam)
{
    RECT rc;
    static UINT cBottom;

    switch (msg) {
    case WM_CREATE:
        GetIniEntries(); /* load strings from STRINGTABLE */
        GetIniSettings(); /* load initialization settings */

        /* Load DIB image. */

        hbmImage = LoadBitmap(hMainInstance, szDIBName);

        /* Create a timer to move the image. */

        idTimer = SetTimer(hwnd, ID_TIMER, wElapsed, NULL);

        xPos = xPosInit;
        yPos = yPosInit;

        break;

    case WM_TIMER:
```

```

    if (fPause && fBottom) {
    if (++cBottom == 10) {
cBottom = 0;
fBottom = FALSE;
    }
    break;
    }

    MoveImage(hwnd); /* move the image slightly */

    break;

    case WM_DESTROY:

        if (hbmImage)
DeleteObject(hbmImage);
        if (idTimer == 0)
KillTimer(hwnd, ID_TIMER);

        break;

        case WM_ERASEBKGD:
GetClientRect(hwnd, &rc);
FillRect((HDC) wParam, &rc,
(HBRUSH) GetObject(BLACK_BRUSH));
        return 0;

        default:
        break;
    }

    return DefScreenSaverProc(hwnd, msg, wParam, lParam);
}

```

If your window procedure traps the WM\_DESTROY message, it must use one of the following methods to properly end the screen saver:

- After processing the message, pass it to the DefScreenSaverProc function.
- In the WM\_DESTROY case of the message handler, call the PostQuitMessage function.

### Configuration Dialog Box

A screen saver uses the ScreenSaverConfigureDialog function to process messages sent to the configuration dialog box. (A screen saver's resource-definition file includes the dialog box template.) The configuration dialog box is displayed when the user selects the Setup button from Desktop section of Control Panel.

The ScreenSaverConfigureDialog function saves its configuration information in the CONTROL.INI file. This configuration information is largely specific to the screen saver and may include such settings as speed, color, number of objects, and position.

The configuration information may also include password protection. When a screen saver is password protected, the user cannot deactivate it and return to the Windows session without typing the password in a dialog box. Adding password protection to a screen saver requires three dialog boxes: one for setting or changing the password, one for typing the password after the screen saver has been activated, and one for informing the user when the password is incorrect. These dialog boxes can be defined as follows:

```

#define ID_OLDTEXT 100
#define ID_NEWTEXT 101
#define ID_AGAIN 102
#define ID_PASSWORD103
#define ID_ETOLD 104
#define ID_ETNEW 105
#define ID_ETAGAIN 106
#define ID_ETPASSWORD 107
#define ID_ICON 108
#define ID_PASSWORDHELP 109

```

```

<>#ifdef RC_INVOKED

DLG_CHANGEPASSWORD DIALOG 8,16,174,79
FONT 8, "MS Sans Serif"
STYLE WS_POPUP | DS_MODALFRAME | WS_CAPTION | WS_SYSMENU
CAPTION "Change Password"
BEGIN
    LTEXT "&Old Password:", ID_OLDTEXT, 4, 3,80,14
    EDITTEXT ID_ETOLD, 84, 3,80,14, ES_PASSWORD
    LTEXT "&New Password:", ID_NEWTEXT, 4,21,80,14
    EDITTEXT ID_ETNEW, 84,21,80,14, ES_PASSWORD
    LTEXT "&Retype New Password:", ID_AGAIN, 4,39,80,14
    EDITTEXT ID_ETAGAIN, 84,39,80,14, ES_PASSWORD
    DEFPUSHBUTTON "OK", IDOK, 4,59,40,14
    PUSHBUTTON "&Help", ID_PASSWORDHELP, 64,59,40,14
    PUSHBUTTON "Cancel", IDCANCEL, 124,59,40,14
END

```

```

DLG_ENTERPASSWORDDIALOG 250,175,170,96
FONT 8, "MS Sans Serif"
STYLE WS_POPUP | DS_MODALFRAME | WS_CAPTION | WS_SYSMENU
CAPTION "<name of screen saver>"
BEGIN
    LTEXT "The screen saver you are using is password protected.
You must type the screen saver password
to turn off the screen saver.", -1, 31,3,140,40
    LTEXT "Password:", ID_PASSWORD, 31,45,40,14
    EDITTEXT ID_ETPASSWORD, 71,45,80,14, ES_PASSWORD
    DEFPUSHBUTTON "OK", IDOK,31,66,40,14
    PUSHBUTTON "Cancel", IDCANCEL,111,66,40,14
    ICON "", ID_ICON, 3, 3,32,32
END

```

```

DLG_INVALIDPASSWORD DIALOG 8,16,174, 79
FONT 8, "MS Sans Serif"
STYLE WS_POPUP | DS_MODALFRAME | WS_CAPTION | WS_SYSMENU
CAPTION "<name of screen saver>"
BEGIN
    ICON "", ID_ICON,3, 3, 0, 0
    LTEXT "Incorrect password;\n\nCheck your screen saver password,
and try again.", -1, 40,3,130,40
    DEFPUSHBUTTON "OK", IDOK, 70,50,40,14
END
<>#endif

```

The preceding example wraps several long lines in the DLG\_ENTERPASSWORD and DLG\_INVALIDPASSWORD dialog boxes. These lines should not wrap in your resource-definition file.

The ScreenSaverConfigureDialog function typically processes a message from a check box that specifies whether the screen saver is password protected and a message specifying that the user has chosen the button to set the password, as shown in the following example:

```

case ID_SETPASSWORD: {
    DLGPROC fpDialog;

    if ((fpDialog = (DLGPROC) MakeProcInstance((FARPROC)
DlgChangePassword,
hMainInstance)) == NULL)
        return FALSE;
    DialogBox(hMainInstance, MAKEINTRESOURCE(DLG_CHANGEPASSWORD),
hWndDlg, fpDialog);
    FreeProcInstance((FARPROC) fpDialog);
    SendMessage(hWndDlg, WM_NEXTDLGCTL, (WPARAM) hWndOK, 11);
}

```

```

        break;
    }

case ID_PASSWORDPROTECTED:
    bPassword ^= 1;
    CheckDlgButton(hwndDlg, wParam, bPassword);
    EnableWindow(hwndSetPassword, bPassword);
    break;

```

The DlgChangePassword function displays the DLG\_CHANGEPASSWORD dialog box.

### Adding Help

The configuration and password dialog boxes for screen savers typically include a Help button. Screen saver applications can check for the Help-button identifier and call the WinHelp function in the same way Help is provided in other Windows applications. In addition, SCRNSAVE.LIB includes HelpMessageFilterHookFunction, which posts the MyHelpMessage message whenever the user presses the F1 key while using a screen saver dialog box. A screen saver can check for this message in the ScreenSaverConfigureDialog function, as follows:

```

switch (msg) {
    .
    . /* process messages */
    .

    default:
        if (msg==MyHelpMessage)
            DoLocalHelpFunc();
}

```

### Exporting Functions

A typical module-definition file for a screen saver application might look like this:

```

NAME      BOUNCER

DESCRIPTION 'SCRNSAVE : Bounce a bitmap'

STUB      'WINSTUB.EXE'
EXETYPEWINDOWS

CODE      MOVEABLE DISCARDABLE PRELOAD
DATA      MOVEABLE MULTIPLE PRELOAD

HEAPSIZE  1024
STACKSIZE 4096

EXPORTS
    ScreenSaverProc @1
    ScreenSaverConfigureDialog @2
    DlgChangePassword@3
    DlgGetPassword @4
    DlgInvalidPassword @5
    HelpMessageFilterHookFunction @6

```

The ScreenSaverProc, ScreenSaverConfigureDialog, DlgChangePassword, and HelpMessageFilterHookFunction functions have been discussed earlier in this topic. The screen saver module typically does not make explicit calls to the HelpMessageFilterHookFunction, DlgGetPassword, or DlgInvalidPassword function.

### Functions

This section describes the functions that applications can use to create a screen saver.

## Windows Debugging Version

The debugging version of the Microsoft Windows operating system generates diagnostic messages whenever it encounters an error that would otherwise cause the system to fail. You use the debugging version by itself or in conjunction with a debugger to debug Windows applications and dynamic-link libraries (DLLs). The debugging functions described in this appendix are not available in the retail version of the system: The API elements exist, but they have no effect. However, the retail version of Windows version 3.1 contains parameter-validation capabilities that an application can use with the Tool Helper library (TOOLHELP.DLL) to retrieve system errors and information about invalid parameters. For more information about the Tool Helper library, see [Tool Helper Library](#).

The following topics discuss the use of the debugging version of the Windows:

[Introduction](#)

[Debugging Programs](#)

[Debugging API Elements](#)

[Error Values](#)

[Debugging Messages](#)

[Common Programming Errors](#)

## Debugging Introduction

The debugging version of Windows consists of the executable and symbol files for the GDI, KERNEL, and USER modules. These modules are identical to those provided with standard Windows except that they contain extra code that checks for errors and then reports them.

The best way to use the debugging version of Windows is to install it on a computer you use for testing and debugging and use a second computer for development. Output from the debugging system and debugger can be directed to a debugging terminal.

Developers who write and debug applications on a single computer often place copies of the standard and debugging versions of Windows in separate directories. When they need to switch from one system to the other, they use batch files to copy the appropriate files to the Windows system directory. (Switching between systems is a good idea because the standard Windows system is faster than the debugging version—it is a better environment for compilers and editors.) You can use the installation program supplied for the Microsoft Windows Software Development Kit (SDK) to set up this two-directory system and then use the batch files D2N.BAT and N2D.BAT to switch between the debugging and standard versions of Windows.

## Introduction

The Microsoft Windows System Debugging Log Application (DBWIN.EXE) allows you to see messages produced by the debugging version of Windows even if you are not using a debugging terminal or debugging application. DBWIN.EXE allows you to control the kinds of messages that are displayed and to save your preferences in the WIN.INI file. DBWIN.EXE also provides a feature that allows you to test the performance of your application during out-of-memory failures.

The Microsoft Windows Dr. Watson application detects system and application failures and can store information in a disk file. This program can help you find and fix problems in your applications. For more information about it, see Object Linking and Embedding Overview.

### Logging Debugging Messages

You can log messages to the DBWIN window, to a debugging monitor, or to the device attached to the COM1 port. The Options menu allows you to change the destination of debugging messages.

### Settings Command

Choosing the Settings command from the Options menu produces a dialog box that allows you to control the display of debugging messages produced by the debugging system. This dialog box contains the following check boxes:

Check box	Description
Break	Controls whether and how a message causes a break to the debugger with a stack trace.
Trace	Controls whether certain kinds of informational messages are produced.
Debugging	Controls the kind of debugging features enabled in the system.

Following is a selected list of debugging options:

Option	Meaning
Validate Heap	Check the consistency of global and local heaps before every call to a memory-management function. This option affects the global heap only when it is one of the default start-up settings (that is, when it is saved by choosing the Save Settings command from the File menu). This option affects local heaps only if it is set before the application is started.
Check Free Blocks	Ensure that freed local blocks are not written into. The value 0xFB is written into free blocks and when the heap is validated, a check is performed to ensure that the blocks are still filled with this value. This option works only with local heaps. It must be used with the Validate Heap option.
Buffer Fill	Fill buffers that are passed to Windows functions with the value 0xF9. This option ensures that all of the supplied buffer is writable and helps detect overwrite problems that can occur when the buffer is too small.
Break with INT 3	Break to the debugger with an int 3 instruction, instead of a fatal exit. This option does not display a stack back-trace.

Note: Some applications will not run when the Buffer Fill option is turned on. If the supplied buffer is smaller than the size specified in the count parameter of the calling function, the application data is overwritten.

### Alloc Break Command

The Alloc Break command on the Options menu ensures that an application deals properly with out-of-memory conditions. This command displays a dialog box into which you can enter the module name of your application and the number of memory allocations you want to succeed before subsequent allocations fail.

The system counts each global or local memory allocation performed by your application. When the number of allocations reaches the allocation break count, that allocation and all subsequent allocations fail. Because memory allocations made by the system fail once the break count is reached, calls to certain functions (such as CreateWindow, CreateBrush, and SelectObject) will fail as well. Only allocations made within the context of the application you specify are affected by the allocation break count.

The module name is limited to 8 characters. In some cases the module name may be different from the

filename. (The module name is specified in the module-definition file for the application.) You cannot specify the module name of a DLL.

If you set the break count to zero, no allocation break is set, but the system counts allocations made by the specified application. You can choose the Show Count button to display the current allocation count.

You can set an allocation break before the named application is run. The allocation count is then set to zero and allocations are counted as soon as the application starts. If you run more than one instance of an application, the allocation break applies only to the most recent instance.

The allocation count is also reset to zero when you choose the Set command or the Inc & Set command. You can set an allocation break before performing an operation, to ensure that your application handles the problem effectively, and then choose Inc & Set and repeat the operation, to ensure that the next allocation failure is also handled properly.

### Interpreting Debugging Messages

Windows debugging messages are the primary feature of the debugging version of Windows. These messages identify errors caused by applications and report the type of each error and the information you need to locate the error in your application.

Windows debugging messages have the following form:

FatalExit Code = fatalexit-code

Stack trace:

```
module-name!segment-name:[function-name+]address
.
.
.
```

Abort, Break or Ignore?

The fatalexit-code parameter identifies the type of error. For a complete set of possible error codes, see Error Values.

The stack trace consists of one or more addresses representing a chain of return addresses from the function that detected the error to the application that made the original function call.

Windows displays the "Abort, Break or Ignore?" prompt at the end of each debugging message.

The following variables are found in Windows debugging messages:

Variable	Description
fatalexit-code	Identifies the type of error (a hexadecimal value).
module-name	Specifies the name of the application or of a Windows module (such as USER, GDI, or KERNEL).
segment-name	Specifies the name of a segment in the application or module.
function-name	Specifies the name of a function in the segment.

Note: The segment and function names are available only if a symbol file (.SYM extension) exists for the given module. Otherwise, Windows displays addresses instead of names.

The following example shows a typical debugging message:

```
FatalExit Code = 0x6040
```

```
Stack trace:
```

```
USER!_FFFE:SHOWCURSOR+0389
USER!_MSGBOX:08D7
USER!_FFFE:922D
MYAPP!_TEXT:WINMAIN+001B
MYAPP!_TEXT:__astart+0060
```

```
Abort, Break or Ignore?
```

In this example, the stack trace shows that the `ShowCursor` function in the USER module (USER.EXE) detected the error. The error type is 0x6040. This value is associated with the `ERR_BAD_HWND` constant; it means that the window handle passed to the function is not valid. The MYAPP application



initially called the USER module at the address WINMAIN+001B in its \_TEXT segment. A check of the application code at that location will probably reveal the error.

The "Abort, Break or Ignore?" prompt gives you the opportunity to terminate Windows, pass control to the debugger, or ignore the error. When you receive this prompt, you must type one of the following responses:

Response	Action
A	Terminates Windows, returning control to the MS-DOS prompt or to the debugger (if one was running).
B	Generates a breakpoint interrupt. If no debugger is running, this response terminates Windows as if you had typed A. If a debugger is running, control passes to the debugger as if you had set a breakpoint in the application. In this case, the CS:IP registers point to an int 3 instruction. To continue execution or to enable single-stepping, you must change the IP register to the address of the next instruction.
I	Ignores the error and continues running the application that caused the error.
SPACE or NEWLINE	Directs Windows to redisplay the debugging message. This is helpful if the stack trace for the message is exceptionally long.

Note: Not all debuggers support the same type of stack trace that Windows displays. If you use the B response to enter a debugger that does not support stack tracing, there is no way to regenerate the trace.

## Debugging API Elements

Applications can use the DebugOutput function to display information on either the debugging terminal or the current debugging computer. The function is especially useful for displaying the full details of calls to functions that generate debugging messages.

DebugOutput includes formatting and message-filtering features that are not available with the OutputDebugString function.

Debugging-system options and filters are provided in the WINDEBUGINFO structure. The WINDEBUGINFO structure has the following form:

```
typedef struct tagWINDEBUGINFO {
    UINT    flags; /* valid WINDEBUGINFO members */
    DWORD   dwOptions; /* debugging options*/
    DWORD   dwFilter; /* filter for trace messages */
    char    achAllocModule[8]; /* module for alloc break*/
    DWORD   dwAllocBreak; /* allocs to succeed before break */
    DWORD   dwAllocCount; /* number of successful allocs*/
} WINDEBUGINFO;
```

The values in WINDEBUGINFO can be set and retrieved by using the SetWinDebugInfo and GetWinDebugInfo functions.

You can generate your own debugging messages by using the FatalExit function. This function displays a message that has the same form as a debugging message generated by Windows, using the error value supplied as its only parameter. This function is especially useful for debugging DLLs.

In general, you should remove calls to debugging functions when compiling the final version of your application or library.

### WIN.INI Debugging Options

Applications use the GetWinDebugInfo and SetWinDebugInfo functions to retrieve or set debugging options or filter values at run time. To control the same options and filter values in a system-wide, persistent manner, you can use two entries in the [WINDOWS] section of the WIN.INI file. These entries are DebugOptions and DebugFilter. They have the following form:

```
[WINDOWS]
    DebugOptions = hexadecimal value
    DebugFilter = hexadecimal value
```

The setting for the DebugOptions entry corresponds to the values for the dwOptions member of the WINDEBUGINFO structure. The setting for the DebugFilter entry corresponds to the values for the dwFilter member of WINDEBUGINFO. To determine the proper hexadecimal value for a setting, add the values of the options to be set. For example, to specify DBO\_CHECKHEAP and DBO\_FREEFILL, the setting for the DebugOptions entry would be 0x0021 (0x0001 + 0x0020). For information about the possible values for these options and a full description of the WINDEBUGINFO structure, see the Microsoft Windows Programmer's Reference, Volume 3.

## Error Values

The following table gives the possible error values in a Windows debugging message:

Value	Constant	Meaning
0x0001	ERR_GALLOC	<u>GlobalAlloc</u> failed. This error value is sent by <u>KERNEL</u> .
0x0002	ERR_GREALLOC	<u>GlobalReAlloc</u> failed. This error value is sent by <u>KERNEL</u> .
0x0003	ERR_GLOCK	<u>GlobalLock</u> failed. This error value is sent by <u>KERNEL</u> .
0x0004	ERR_LALLOC	<u>LocalAlloc</u> failed. This error value is sent by <u>KERNEL</u> .
0x0005	ERR_LREALLOC	<u>LocalReAlloc</u> failed. This error value is sent by <u>KERNEL</u> .
0x0006	ERR_LLOCK	<u>LocalLock</u> failed. This error value is sent by <u>KERNEL</u> .
0x0007	ERR_ALLOCRES	<u>AllocResource</u> failed. This error value is sent by <u>KERNEL</u> .
0x0008	ERR_LOCKRES	<u>LockResource</u> failed. This error value is sent by <u>KERNEL</u> .
0x0009	ERR_LOADMODULE	<u>LoadModule</u> failed. This error value is sent by <u>KERNEL</u> .
0x0040	ERR_CREATEDLG	Dialog box could not be created because <u>LoadMenu</u> failed. This error value is sent by USER.
0x0041	ERR_CREATEDLG2	Dialog box could not be created because <u>CreateWindow</u> failed. This error value is sent by USER.
0x0042	ERR_REGISTERCLASS	<u>RegisterClass</u> failed because the class is already registered. This error value is sent by USER.
0x0043	ERR_DCBUSY	Device-context cache is full. This error value is sent by USER.
0x0044	ERR_CREATEWND	Window could not be created because the class was not found. This error value is sent by USER.
0x0045	ERR_STRUCEXTRA	Program is using unallocated space. This error value is sent by USER.
0x0046	ERR_LOADSTR	<u>LoadString</u> failed. This error value is sent by USER.
0x0047	ERR_LOADMENU	<u>LoadMenu</u> failed. This error value is sent by USER.
0x0048	ERR_NESTEDBEGINPAINT	Program contains nested <u>BeginPaint</u> functions. This error value is sent by USER.
0x0049	ERR_BADINDEX	Index to <u>GetClassLong</u> , <u>GetClassWord</u> , <u>GetWindowLong</u> , <u>GetWindowWord</u> , <u>SetClassLong</u> , <u>SetClassWord</u> , <u>SetWindowLong</u> , or <u>SetWindowWord</u> is invalid. This error value is sent by USER.
0x004A	ERR_CREATEMENU	Menu could not be created. This error value is sent by USER.
0x0080	ERR_CREATEDC	<u>CreateCompatibleDC</u> , <u>CreateDC</u> , or <u>CreateIC</u> failed. This error value is sent by GDI.
0x0081	ERR_CREATEMETA	<u>CreateMetaFile</u> failed. This error value is sent by GDI.
0x0082	ERR_DELOBJSELECTED	Program is trying to delete a bitmap that is selected into the device context. This error value is sent by GDI.
0x0083	ERR_SELBITMAP	Program is trying to select a bitmap that is already selected. This error value is sent by GDI.
0x6001	ERR_BAD_VALUE	A 16-bit signed or unsigned value is invalid.
0x6002	ERR_BAD_FLAGS	One or more bit flags are invalid.

0x6003	ERR_BAD_INDEX	Index is invalid or out of range.
0x6009	ERR_BAD_SELECTOR	Selector is invalid.
0x600B	ERR_BAD_HANDLE	Generic handle is invalid.
0x6020	ERR_BAD_HINSTANCE	Instance handle is invalid. This error value is sent by KERNEL.
0x6021	ERR_BAD_HMODULE	Module handle is invalid. This error value is sent by KERNEL.
0x6022	ERR_BAD_GLOBAL_HANDLE	Global handle is invalid. This error value is sent by KERNEL.
0x6023	ERR_BAD_LOCAL_HANDLE	Local handle is invalid. This error value is sent by KERNEL.
0x6024	ERR_BAD_ATOM	Atom is invalid. This error value is sent by KERNEL.
0x6025	ERR_BAD_HFILE	File handle is invalid. This error value is sent by KERNEL.
0x6040	ERR_BAD_HWND	Window handle is invalid. This error value is sent by USER.
0x6041	ERR_BAD_HMENU	Menu handle is invalid. This error value is sent by USER.
0x6042	ERR_BAD_HCURSOR	Cursor handle is invalid. This error value is sent by USER.
0x6043	ERR_BAD_HICON	Icon handle is invalid. This error value is sent by USER.
0x6044	ERR_BAD_HDWP	Handle to a window-position structure is invalid. This error value is sent by USER.
0x6045	ERR_BAD_CID	Communications identifier (CID) is invalid. This error value is sent by USER.
0x6046	ERR_BAD_HDRV	Installable-driver handle is invalid. This error value is sent by USER.
0x6061	ERR_BAD_GDI_OBJECT	GDI object is invalid. This error value is sent by GDI.
0x6062	ERR_BAD_HDC	Device-context handle is invalid. This error value is sent by GDI.
0x6063	ERR_BAD_HPEN	Pen handle is invalid. This error value is sent by GDI.
0x6064	ERR_BAD_HFONT	Font handle is invalid. This error value is sent by GDI.
0x6065	ERR_BAD_HBRUSH	Brush handle is invalid. This error value is sent by GDI.
0x6066	ERR_BAD_HBITMAP	Bitmap handle is invalid. This error value is sent by GDI.
0x6067	ERR_BAD_HRGN	Region handle is invalid. This error value is sent by GDI.
0x6068	ERR_BAD_HPALETTE	Palette handle is invalid. This error value is sent by GDI.
0x6069	ERR_BAD_HMETAFILE	Metafile handle is invalid. This error value is sent by GDI.
0x7004	ERR_BAD_DVALUE	A 32-bit signed or unsigned value is invalid.
0x7005	ERR_BAD_DFLAGS	One or more 32-bit flags are invalid.
0x7006	ERR_BAD_DINDEX	A 32-bit index is invalid or out of range.
0x7007	ERR_BAD_PTR	Pointer is invalid.
0x7008	ERR_BAD_FUNC_PTR	Function pointer is invalid.
0x700A	ERR_BAD_STRING_PTR	Zero-terminated string pointer is invalid.
0x7060	ERR_BAD_COORDS	X- and y-coordinates are invalid. This error value is sent by GDI.

The following error values may have been combined with other values in the preceding table to identify the type of error:

Value	Constant	Meaning
0x4000	ERR_PARAM	Parameter is invalid. This flag is always set for parameter-validation error messages.
0x8000	ERR_WARNING	Nonfatal error occurred. An invalid parameter was detected, but the error was not serious enough to cause the function to fail. The invalid parameter is reported, but the function executes as usual.

To determine the size of an invalid parameter, you can combine `ERR_SIZE_MASK` (0x3000) with other error values by using the AND operator. The following table gives the possible results of this operation:

Value	Constant	Meaning
0x1000	ERR_BYTE	An 8-bit parameter is invalid.
0x2000	ERR_WORD	A 16-bit parameter is invalid.
0x3000	ERR_DWORD	A 32-bit parameter is invalid.

## Debugging Messages

The following table gives the strings that are displayed as Windows debugging messages:

**Activation failed: system modal window is present**  
Windows may not be activated by another application while a system modal window is present. This warning message is sent by the USER module.

**Alloc break: Failing allocation**  
All further memory allocations will fail; this supports the Alloc Break debugging setting. This error message is sent by the KERNEL module.

**AllocResource failed**  
The AllocResource function failed, probably due to insufficient memory. This warning message is sent by the KERNEL module.

**App not initialized**  
A function was called before the application was properly initialized. This error message is sent by the USER module.

**application-name Automatic Data Segment larger than 64K.**  
The application's automatic data segment, combined with the heap and stack, must be less than 64K. This error message is sent by the KERNEL module.

**application-name Compacting heap, discarding segments**  
An application has specified a negative value in a call to the GlobalCompact function, forcing the system to free all of the code segments that have been loaded from disk. Instead, applications should simply allocate any needed memory and allow the system to compact memory as required. This trace message is sent by the KERNEL module to help applications optimize memory management.

**application-name failed implicit link to module-name**  
When an application calls the address specified by this warning, the system forces a call to the FatalAppExit function. This warning occurs when the application starts. An application will sometimes call the specified address after verifying that all entry points are valid. (For example, an application might verify the Windows version before calling functions that exist only in Windows 3.1.) This warning message is sent by the KERNEL module.

**application-name: reading resource value1 value2**  
The system is reading resources from disk. This trace message is sent by the KERNEL module to help applications optimize loading and execution.

**Attempt to activate destroyed window**  
A window was activated during processing of the WM\_DESTROY or WM\_NCDESTROY message. No active window is produced. This error message is sent by the USER module.

**Attempt to delete object still selected in SaveDC stack**  
An application attempted to delete an object that was still in use because of an earlier call to the SaveDC function. Deleting an object that is still saved can cause the system to crash. Applications must call the RestoreDC function and select the object out of the device context before deleting the object. This error message is sent by the GDI module.

**Bad GWW\_/GWL\_/GCW\_ index value**  
An invalid negative index was used in a call to the GetWindowWord, SetWindowWord, GetClassWord, or SetClassWord function. This warning can occur in the retail version of Windows. The LogError constant is ERR\_BADINDEX. This warning message is sent by the USER module.

**Beginning app termination cleanup...**  
The system application-termination routine has begun. This trace message is sent by the USER module.

**BOOT: unable to load filename**  
A startup error occurred. This error message is sent by the KERNEL module.

**BS\_USERBUTTON no longer supported**  
The BS\_USERBUTTON style is no longer supported in Windows 3.1. Use BS\_OWNERDRAW instead. This warning message is sent by the USER module.

**Can't change WS\_EX\_TOPMOST with SetWindowLong**  
An attempt was made to change the WS\_EX\_TOPMOST style by using the SetWindowLong function. The style bit is not changed. Use the SetWindowPos function's HWND\_TOPMOST or HWND\_NOTOPMOST values to change this flag. This error message is sent by the USER module.

**Can't find filename**  
A startup error occurred. This error message is sent by the KERNEL module.

**Can't load segment.**  
A disk or link error occurred. This error message is sent by the KERNEL module.

Can't post system error dialog: app not initialized

A system error occurred before an application was initialized. This sometimes happens because of an error during the initialization of a DLL. This warning message is sent by the USER module.

Clipboard already open

The OpenClipboard function was called when the clipboard was already open. This error message is sent by the USER module.

CreateDialog() failed: couldn't create control

The creation of a dialog box from a dialog-box template failed because a control could not be created. This warning can occur in the retail version of Windows. The LogError constant is ERR\_CREATEDLG. This warning message is sent by the USER module.

CreateDialog() failed: couldn't create window

The creation of a dialog box from a dialog-box template failed because the dialog box could not be created. This warning can occur in the retail version of Windows. The LogError constant is ERR\_CREATEDLG2. This warning message is sent by the USER module.

CreateDialog() failed: couldn't load menu

The creation of a dialog box from a dialog-box template failed because the menu resource in the template could not be created. This warning can occur in the retail version of Windows. The LogError constant is ERR\_CREATEDLG. This warning message is sent by the USER module.

CreateMenu failed

The CreateMenu function failed because of a memory shortage. This warning can occur in the retail version of Windows. The LogError constant is ERR\_CREATEMENU. This warning message is sent by the USER module.

CreateWindow failed: Out of memory

The CreateWindow function failed because of a memory shortage. This warning can occur in the retail version of Windows. The LogError constant is ERR\_CREATEWND. This warning message is sent by the USER module.

CreateWindow failed: Window class not found

The CreateWindow function was called with a nonexistent window class name. This error can occur in the retail version of Windows. The LogError constant is ERR\_CREATEWND. This error message is sent by the USER module.

CreateWindow(): Invalid parent hwnd

A child window was created without a valid parent window handle. This error can occur in the retail version of Windows. The LogError constant is ERR\_CREATEWND. This error message is sent by the USER module.

CreateWindow(): NULL instance handle

The CreateWindow function was called with a NULL instance handle. This error can occur in the retail version of Windows. The LogError constant is ERR\_CREATEWND. This error message is sent by the USER module.

CreateWindow: Out of memory

The WM\_NCCREATE message returned FALSE, preventing the window from being destroyed. This warning message is sent by the USER module.

Data Segment N of module-name can't be discardable

Certain segments of an application or DLL must be preloaded. If an application sets the switch incorrectly, the system corrects the setting. However, the application will load faster if it sets the switch properly. The following segments must be preloaded: DATA segments, FIXED CODE segments, and CODE segments that are MOVEABLE but not DISCARDABLE. This warning message is sent by the KERNEL module.

DC Cache full: Too many GetDC() calls

More than five nested calls to the GetDC function were made without subsequent calls to the ReleaseDC function. This situation can cause system deadlock; device contexts must be released before further calls to GetDC. This error can occur in the retail version of Windows. The LogError constant is ERR\_DCBUSY. This error message is sent by the USER module.

DecExeUsage(application-name) not DLL

A low-memory situation can cause this error. This error message is sent by the KERNEL module.

Default Data Segment of module-name must be preload

Certain segments of an application or DLL must be preloaded. If an application sets the switch incorrectly, the system corrects the setting. However, the application will load faster if it sets the switch properly. The following segments must be preloaded: DATA segments, FIXED CODE segments, and CODE segments that are MOVEABLE but not DISCARDABLE. This warning message is sent by the KERNEL module.

DeferWindowPos: All windows must share same parent

- All windows positioned by using the DeferWindowPos function must be siblings. This error message is sent by the USER module.
- DefMDIChildProc called on a non-MDIChild window**  
The DefMDIChildProc function was called with a window that is not a proper MDI child window. This error message is sent by the USER module.
- Demand load module-name(N) on application-name**  
The system is demand-loading the specified segment of the specified module. If an application loads many segments when starting, the application should probably add these segments to the FastLoad block. (To add a segment to the FastLoad block, give the segment the PRELOAD attribute.) This change does not affect total memory requirements. If there is not enough memory to load all the PRELOAD segments, they won't be preloaded. This trace message is sent by the KERNEL module to help applications optimize loading and execution.
- DestroyCursor: Destroying current cursor**  
The DestroyCursor function was called with the currently-selected cursor. The arrow cursor was selected in its place. This error message is sent by the USER module.
- DestroyWindow: hwnd not created by the current task**  
The DestroyWindow function was called with a window created by another application. An application cannot destroy windows created by other applications. This error message is sent by the USER module.
- DestroyWindow: System menu handle no longer valid**  
A window's system menu window was destroyed. Normally, applications should never destroy the system menu handle; this handle is automatically destroyed when the window is destroyed. This warning message is sent by the USER module.
- DestroyWindow: Unremoved window property**  
A window was destroyed without removing all of its properties. Properties should be removed before the window is destroyed or during the processing of the WM\_DESTROY message. This warning message is sent by the USER module.
- DestroyWindow: Window menu no longer valid**  
A window menu was destroyed without clearing the handle by calling the SetMenu function with NULL as the second parameter. The menu associated with a window is destroyed when the window is destroyed. If you destroy the menu separately, always use SetMenu to clear the handle. This warning message is sent by the USER module.
- Dialog class registered with cbWndExtra < DLGWINDOWEXTRA**  
The DefDlgProc function was called with a window whose class was not registered with the proper value for the cbWndExtra member of the WNDCLASS structure. Windows, when used with the dialog manager, must use the DLGWINDOWEXTRA constant for this member. This error message is sent by the USER module.
- Dialog control id not found**  
An invalid dialog-control identifier was passed to one of the dialog-box functions that take a dialog-box handle and control identifier (for example, the SetDlgItemText function). This warning message is sent by the USER module.
- Dialog should be dismissed with EndDialog, not DestroyWindow**  
The dialog window was destroyed during the processing loop for dialog messages. The EndDialog function should be used instead. This error message is sent by the USER module.
- Dialog window destroyed in dialog callback**  
The dialog window was destroyed by the dialog function. The EndDialog function should be used to destroy a dialog window. This error message is sent by the USER module.
- Dialog window owner destroyed while dialog still valid**  
The owner of a dialog box window was destroyed before the dialog box was destroyed. This can be avoided by calling the SetWindowWord function with GW\_W\_HWNDPARENT set to NULL. This warning message is sent by the USER module.
- Discardable temp buffer busy**  
An internal error has corrupted the internal state of GDI. (This error should never occur.) This error message is sent by the GDI module.
- Divide by zero or divide overflow error: break and trace till IRET**  
A divide-by-zero or divide-overflow fault has occurred. To find the fault, trace with your debugger to the IRET instruction: one more trace will take you to the instruction that caused the error. This fatal error message is sent by the USER module.
- DlgDirList: id not a list box or combo box**  
The DlgDirList function was called with a control identifier specifying a control other than a list-box or combo-box class window. This error message is sent by the USER module.



DlgDirList: list box or combo box id not found

The `DlgDirList` function was called with a control identifier specifying a nonexistent control. This error message is sent by the USER module.

Edit SetText: 3.0 compat AnsiUpper being done on source text

The application is using the `AnsiUpper` function unnecessarily. For single-line edit controls with the `ES_UPPERCASE` style, the string that is passed to `WM_SETTEXT` and other messages is not typically modified by edit control code, but is converted to uppercase internally. This corresponds to the WIN.INI [Compatibility] section, value 0x0080. This error message is sent by the USER module.

End of app termination cleanup

The system-application termination routine has finished. This trace message is sent by the USER module.

Error reading relocation records from module-name

I/O error loading a segment. This error message is sent by the KERNEL module.

Error value loading filename

The `LoadModule` function failed. The message displays the `LoadModule` return value and the name of the file being loaded. This warning message is sent by the KERNEL module.

ES\_READONLY not supported in 3.0 edit ctrls: use EM\_SETREADONLY

The `ES_READONLY` edit control style was specified by a 3.0 application. Use `EM_SETREADONLY` instead. This error message is sent by the USER module.

Exiting menu mode: another window activated

A menu was canceled because another window in the system was being activated. This warning can occur if a dialog box or message box is brought up while a menu is displayed. This warning message is sent by the USER module.

FastLoad area ignored due to incorrect segment flags

If the system must change the `PRELOAD` flag of a segment, it invalidates the `FastLoad` block and the application is loaded relatively slowly. This warning message is sent by the KERNEL module.

Fault detected - handled by module-name segment:offset

A general-protection fault occurred in a `WEP` (Windows exit procedure). The system continues operations with the next step. This error message is sent by the KERNEL module.

Fault in SegReloc value1 value2

A disk or link error occurred. This error message is sent by the KERNEL module.

GDI: %s not deleted: %04X

An application has neglected to delete certain GDI objects upon termination. These objects are not deleted by GDI; in order to avoid using up system resources, an application must delete all GDI objects that it creates. This message can also occur with certain objects that an application uses that were created by a DLL. When a DLL creates an object to share among multiple applications, this message may result when the first application that uses the DLL terminates, even though the object should not be deleted at that time. This warning message is sent by the GDI module.

GDI: Attempt to delete object owned by system

An application attempted to delete an object that is owned by the system. For example, this error occurs when a window class brush is deleted by the application after the class has been registered. This error also occurs when an application reuses a deleted object handle. This error message is sent by the GDI module.

GDI: DeleteObject:%s(%04X) selection count incorrect

The internal state of GDI has been corrupted and the system may be in an unstable state. This warning can occur because of writing through uninitialized pointers or other programming errors and is usually a symptom of a more serious problem elsewhere. This warning message is sent by the GDI module.

GDI: DeleteObject:%s(%04X) still selected in DC(s)

An application attempted to delete an object that was still selected in a device context. Applications must always deselect objects before deleting them. If an application deletes an object that is selected into a device context and then attempts to draw in the device context, the system may be left in an indeterminate state. This warning message is sent by the GDI module.

GDI: Unable to deselect %04X

The internal state of GDI has been corrupted and the system may be in an unstable state. This warning can occur because of writing through uninitialized pointers or other programming errors and is usually a symptom of a more serious problem elsewhere. This warning message is sent by the GDI module.

Get file offset failed

A disk or link error occurred. This error message is sent by the KERNEL module.

GetAtomName(0xNNNN,...) Can't find atom  
An atom was not found in a call to the GetAtomName function. This warning message is sent by the KERNEL module.

GetDC without ReleaseDC  
A window was destroyed before calling the ReleaseDC function. This error message is sent by the USER module.

GetDCEX: Can't find permanent DC  
The GetDCEX function was called for a window that does not have the CS\_OWNDC or CS\_CLASSDC style without setting the DCX\_CACHE flag. This warning message is sent by the USER module.

GetMenu: Window menu no longer valid  
A window's menu was previously destroyed, and the menu returned by the GetMenu function is no longer valid. If a window menu is destroyed, the SetMenu function should be called with the second parameter set to NULL to clear the handle stored in the window. This warning message is sent by the USER module.

GetNextDriver: Invalid starting driver handle  
The GetNextDriver function was called with an invalid driver handle. This error message is sent by the USER module.

Global class freed with existent class windows!  
A global class registered by a DLL is being freed while windows of that class still exist. This serious error is usually caused by the incorrect termination of an application's DLL. This error message is sent by the USER module.

GlobalAlloc failed  
The GlobalAlloc function failed, probably due to insufficient memory. This warning message is sent by the KERNEL module.

GlobalAlloc(0xNNNNNNN) failed for application-name  
A call to the GlobalAlloc function failed. This typically happens because the requested memory is too large. This trace message is sent by the KERNEL module to help applications optimize memory management.

GlobalReAlloc failed  
The GlobalReAlloc function failed, probably due to insufficient memory. This warning message is sent by the KERNEL module.

GlobalWire(N of module-name) (try GlobalLock)  
Applications should generally use the GlobalLock function to lock memory instead of the GlobalWire function. GlobalWire should not be used in Windows 3.1. This warning message is sent by the KERNEL module.

GP fault in \_hread/\_hwrite at value1 value2  
A general-protection fault occurred while reading or writing a huge file. This is a user error. This error message is sent by the KERNEL module.

GP fault in LStrNCpy  
A general-protection fault occurred when copying a string. This error message is sent by the KERNEL module.

greserve doesn't fit  
Memory is low. This error message is sent by the KERNEL module.

greserve: 0xNNNNNN bytes  
Memory has been reserved by the system for discardable segments. This trace message is sent by the KERNEL module to help applications optimize memory management.

GrowHeap: 0xNNNNNN allocated  
Memory has been allocated from a DPMI server (Win386 or DOSX). This trace message is sent by the KERNEL module to help applications optimize memory management.

hMemCopy: Copy past end of segment  
A general-protection fault occurred while copying a huge memory block. This is a user error. This error message is sent by the KERNEL module.

Hook Not Allowed  
An application attempted to install a task-specific hook when only a system hook was allowed, such as WH\_JOURNALRECORD, WH\_JOURNALPLAYBACK, or WH\_SYSMSGFILTER. This error message is sent by the USER module.

IncExeUsage(application-name) not DLL  
A low-memory situation can cause this error. This error message is sent by the KERNEL module.

IncExeUsage: ne\_usage overflow

An internal error occurred. This fatal error message is sent by the KERNEL module.

**Intertask SendMessage() during app termination**  
An inter-application call to the SendMessage function occurred during application termination. Usually this means that an application failed to destroy all of its windows before terminating. This warning message is sent by the USER module.

**Intertask SendMessage() not allowed: Tasks locked**  
An inter-application call to the SendMessage function was attempted while a system modal dialog box was displayed, or the system has locked all but the current task. Using SendMessage to send messages to other applications is not allowed while a system modal dialog box is displayed. This error message is sent by the USER module.

**Intertask SendMessage: Sleeping since unreplied SendMessage pending**  
An application attempted to send a message to another application before the second application processed an earlier inter-application call to the SendMessage function. This warning indicates that there may be a hung application in the system. This warning message is sent by the USER module.

**Invalid button style**  
An invalid button class style was supplied. This error message is sent by the USER module.

**Invalid clipboard metafile**  
An invalid metafile handle was placed in the clipboard by using the SetClipboardData function. This error message is sent by the USER module.

**Invalid color index.**  
An invalid color index was specified in a call to the SetSysColors function. This error message is sent by the USER module.

**Invalid driver entry proc address**  
An installable driver entry procedure was not declared with the PASCAL keyword or was otherwise improperly implemented. The driver will not be installed. This error message is sent by the USER module.

**Invalid EXE file filename**  
A startup error occurred. This error message is sent by the KERNEL module.

**Invalid function called: System state potentially trashed**  
The edit control window procedure is being called by an application. Applications should always use the CallWindowProc function and the previous window procedure address returned either from a call to the GetClassInfo function or a call to the GetWindowLong function using the GWL\_WNDPROC constant. This error message is sent by the USER module.

**Invalid HBRUSH returned by WM\_CTLCOLOR message**  
An invalid brush handle was returned by the WM\_CTLCOLOR message. This error message is sent by the USER module.

**Invalid Hook Code**  
An invalid negative hook code value was passed to the DefHookProc or DefHookProcEx function. This error message is sent by the USER module.

**Invalid Hook Handle**  
An invalid hook handle was passed to the DefHookProcEx or UnhookWindowsHookEx function. This error message is sent by the USER module.

**Invalid Hook ID**  
An invalid hook identifier was passed to the SetWindowsHook or SetWindowsHookEx function. This error message is sent by the USER module.

**Invalid Hook Instance**  
An invalid hook instance handle was passed to the SetWindowsHookEx function. This error message is sent by the USER module.

**Invalid Hook Proc Addr**  
The hook function address specified in a call to the SetWindowsHook function is invalid. This error message is sent by the USER module.

**Invalid ordinal reference (#NNN) to application-name**  
When an application calls the address specified by this warning, the system forces a call to the FatalAppExit function. This warning occurs when the application starts. An application will sometimes call the specified address after verifying that all entry points are valid. (For example, an application might verify the Windows version before calling functions that exist only in Windows 3.1.) This warning message is sent by the KERNEL module.

**Invalid protect mode EXE file filename**  
A startup error occurred. This error message is sent by the KERNEL module.

**Invalid segment in fixup.**

A disk or link error occurred. This error message is sent by the KERNEL module.

**Invalid ShowWindow command**  
An invalid command was specified in a call to the ShowWindow function. This error message is sent by the USER module.

**Invalid size for DRIVERINFOSTRUCT**  
The length member of the DRIVERINFOSTRUCT structure was not properly initialized when the structure was passed to the GetDriverInfo function. This member should be set to the size of the DRIVERINFOSTRUCT structure. This error message is sent by the USER module.

**Invalid SPI\_\* parameter**  
The SystemParametersInfo function was called with an invalid uAction parameter. This error message is sent by the USER module.

**Invalid task handle**  
An invalid task handle was passed to the SetWindowsHookEx function. This error message is sent by the USER module.

**Invalidation with fErase==FALSE prevents WM\_ERASEBKGDND**  
In Windows 3.1, a call to the InvalidateRect function does not prevent pending WM\_ERASEBKGDND messages from being sent. In Windows 3.0, if InvalidateRect was called with the lprc parameter equal to NULL and the fErase parameter equal to FALSE, any pending WM\_ERASEBKGDND messages were validated and were not sent. This warning message is sent by the USER module.

**KReboot: Trying to look up application-name**  
The system disables the local reboot capability for known modules. This informational trace message is sent by the KERNEL module.

**Loading filename**  
The name of the application, DLL, or driver is now being loaded. This informational trace message is sent by the KERNEL module.

**Loading module-name Nonresident name table**  
The system loads the nonresident-name table when an application specifies a name in a call to the GetProcAddress function and the DLL does not have the name in the resident-name table. When a name is not in the resident-name table, it is either in the nonresident-name table or the name does not exist. This message demonstrates the performance decrease that occurs whenever the nonresident-name table is loaded. This trace message is sent by the KERNEL module to help applications optimize loading and execution.

**LoadString() failed**  
The LoadString function failed because a resource could not be found. This warning can occur in the retail version of Windows. The LogError constant is ERR\_LOADSTR. This warning message is sent by the USER module.

**Local free memory overwritten at segment:offset**  
The local heap has been corrupted. This fatal error message is sent by the KERNEL module.

**LocalAlloc failed**  
The LocalAlloc function failed, probably due to insufficient memory. This warning message is sent by the KERNEL module.

**LocalLock failed**  
The LocalLock function failed, probably due to an invalid handle or a corrupted heap. This warning message is sent by the KERNEL module.

**LocalReAlloc failed**  
The LocalReAlloc function failed, probably due to insufficient memory. This warning message is sent by the KERNEL module.

**LockInput called with input already unlocked**  
The LockInput function was called to unlock input when the input was already unlocked. This error message is sent by the USER module.

**LockInput() called when already locked**  
The LockInput function was called when the input was already locked by a previous call to LockInput. This error message is sent by the USER module.

**LockResource failed**  
The LockResource function failed, probably due to an invalid resource handle or insufficient memory. This warning message is sent by the KERNEL module.

**looking for entry-name**  
The system loads the nonresident-name table when an application specifies a name in a call to the GetProcAddress function and the DLL does not have the name in the resident-name table. When a

name is not in the resident-name table, it is either in the nonresident-name table or the name does not exist. This message demonstrates the performance decrease that occurs whenever the nonresident-name table is loaded. This trace message is sent by the KERNEL module to help applications optimize loading and execution.

**MakeProcInstance failed. Did you check return values?**  
A call to the MakeProcInstance function failed. This error message is sent by the KERNEL module.

**Menu destroyed unexpectedly by WM\_INITMENU**  
The DestroyMenu function was called unexpectedly during processing of the WM\_INITMENU message. This error message is sent by the USER module.

**Menu destroyed unexpectedly by WM\_INITMENUPOPUP**  
The DestroyMenu function was called unexpectedly during processing of the WM\_INITMENUPOPUP message. This error message is sent by the USER module.

**MessageBox failed: app not initialized**  
The MessageBox function was called (or an error occurred) before an application was properly initialized. This error sometimes happens during the initialization of a DLL. This error message is sent by the USER module.

**Metafile has incorrect size**  
The data contained in a metafile is invalid. This warning message is sent by the GDI module.

**Metafile is not terminated properly**  
The data contained in a metafile is invalid. This warning message is sent by the GDI module.

**Missing BeginPaint() or GetUpdateRect/Rgn(fErase == TRUE) in WM\_PAINT**  
A WM\_PAINT message was handled incorrectly. In order to ensure that all necessary WM\_NCPAINT messages are sent properly, a window that processes a WM\_PAINT message must call the BeginPaint function or call either the GetUpdateRect or GetUpdateRgn function with the fErase parameter equal to TRUE. This warning message is sent by the USER module.

**Module Name module-name (application-name) too long**  
The names of application modules are limited to 8 bytes. This warning message is sent by the KERNEL module.

**Module unloaded with windows still subclassed**  
A DLL was terminated or unloaded while a window in the system was subclassed with a function defined in that DLL. This is a serious error. In general, DLLs that contain subclassed functions must not be freed unless all windows that may have been subclassed with that function are destroyed. This error message is sent by the USER module.

**module-name has invalid relocation record**  
A disk or link error occurred. This error message is sent by the KERNEL module.

**module-name I/O error reading segment**  
An I/O error in loading a segment occurred. This error message is sent by the KERNEL module.

**module-name MakeProcInstance only for current instance.**  
An application is calling the MakeProcInstance function incorrectly. This fatal error message is sent by the KERNEL module.

**module-name segment:offset called undefined dynalink**  
The specified module attempted to link to a function exported by a DLL, but the system could not find the function. The application attempted to call the entry point even though the function was not found. This fatal error message is sent by the KERNEL module.

**Multiple properties removed during enumerate**  
More than one window property was removed during a property-enumeration callback function; there may be improper property enumeration. Generally, only the enumerated window property may be removed during the enumeration callback function. This warning message is sent by the USER module.

**MyOpenFile not reentrant**  
An internal error occurred. This error message is sent by the KERNEL module.

**Nested BeginPaint() calls**  
The BeginPaint function was called a second time for a window before the EndPaint function was called. This warning usually occurs when an application calls a function such as UpdateWindow during the processing of a WM\_PAINT message. This should be avoided because it may cause incorrect clipping regions in the device context after a call to EndPaint. This warning can occur in the retail version of Windows. The LogError constant is ERR\_NESTEDBEGINPAINT. This warning message is sent by the USER module.

**not enough stack space for DX array. String truncated.**

- An application has called the TextOut or ExtTextOut function with a very long string (> 2048 characters); there was not enough stack space for temporary storage. This problem is typically solved by breaking the string up into shorter strings, although an application with insufficient stack space may encounter this warning with smaller strings. This warning message is sent by the GDI module.
- NULL handle.  
A disk or link error occurred. This error message is sent by the KERNEL module.
- NULL segment in fixup.  
A disk or link error occurred. This error message is sent by the KERNEL module.
- Obsolete function ControlPanelInfo() called  
The obsolete ControlPanelInfo function was called. Use the SystemParametersInfo function instead. This error message is sent by the USER module.
- Obsolete function GetInternalWindowPos() called  
The obsolete GetInternalWindowPos function was called. Use the GetWindowPlacement function instead. This error message is sent by the USER module.
- Obsolete function SetDeskPattern called: use SystemParametersInfo  
The obsolete SetDeskPattern function was called. Use the SystemParametersInfo function instead. This error message is sent by the USER module.
- Obsolete function SetDeskWallPaper() called  
The obsolete SetDeskWallPaper function was called. Use the SystemParametersInfo function instead. This error message is sent by the USER module.
- Obsolete function SetInternalWindowPos() called  
The obsolete SetInternalWindowPos function was called. Use the SetWindowPlacement function instead. This error message is sent by the USER module.
- Out of files (set FILES=30 in CONFIG.SYS) filename  
A startup error occurred. This error message is sent by the KERNEL module.
- Out of mem loading seg module-name  
This internal error should never occur. This error message is sent by the KERNEL module.
- Popup menu incorrectly activated by application  
The ActivateWindow function was called with the handle of a pop-up menu. The system will activate the owner of the pop-up window instead. This error message is sent by the USER module.
- Read record failed  
A disk or link error occurred. This error message is sent by the KERNEL module.
- Read value bytes, expecting value.  
A disk or link error occurred. This error message is sent by the KERNEL module.
- Reentrant application termination  
The system application-termination routine was reentered; that is, two applications were terminating at the same time. This situation can occur if an application is terminated by another when it is processing activation messages, during DLL WEP function processing or installable driver DRV\_EXITAPPLICATION message processing. This reentrancy situation can result in timing errors that are difficult to debug. This warning message is sent by the USER module.
- RegisterClass failed: class already exists  
A window class was registered with a name that has already been registered. This warning can occur in the retail version of Windows. The LogError constant is ERR\_REGISTERCLASS. This warning message is sent by the USER module.
- RegisterClass failed: global class already exists  
A global window class was registered with a name that already has been registered. This error message is sent by the USER module.
- RegisterClass failed: out of memory  
The RegisterClass function failed because of a memory shortage. This error message is sent by the USER module.
- RegisterClass: HACK! Fixing up bogus cbWndExtra and cbClsExtra  
An extra 4 bytes of window and class word space is added to all window classes created by this application. This addition fixes a serious problem in some applications using unallocated window words. This corresponds to the WIN.INI [Compatibility] section, value 0x0100. This error message is sent by the USER module.
- RegisterClass: Invalid class brush  
The hbrBackground member of the WNDCLASS structure is invalid. This error message is sent by the USER module.
- RegisterClass: Invalid class style

The style member of the WNDCLASS structure is invalid. This error message is sent by the USER module.

RegisterClass: Invalid HINSTANCE  
The hInstance member of the WNDCLASS structure was NULL or otherwise invalid. This error message is sent by the USER module.

RegisterClass: Negative cbClsExtra  
The cbClsExtra member of the WNDCLASS structure contains a negative number. This error message is sent by the USER module.

RegisterClass: Negative cbWndExtra  
The cbWndExtra member of the WNDCLASS structure contains a negative number. This error message is sent by the USER module.

RegisterClass: NULL window proc  
The lpfnWndProc member of the WNDCLASS structure does not contain a valid function pointer. This error message is sent by the USER module.

RegisterClass: Unusually large cbClsExtra (> 40)  
The cbClsExtra member of the WNDCLASS structure contains a number that is too large. This value should be less than 40. This is an error message if the application is running with Windows 3.1; otherwise, it is a warning message. It is sent by the USER module.

RegisterClass: Unusually large cbWndExtra (> 40)  
The cbWndExtra member of the WNDCLASS structure contains a number that is too large. In order to avoid using system resources, applications should limit the number of extra window words to less than 40 bytes, and preferably to less than 10 bytes. It is usually best to store a single pointer to a private data structure that is allocated elsewhere. This is an error message if the application is running with Windows 3.1; otherwise, it is a warning message. It is sent by the USER module.

RegisterClass: Window proc not exported  
The lpfnWndProc member of the WNDCLASS structure contains a pointer to a function that is not properly exported. This error message is sent by the USER module.

ReleaseDC: DC already released  
A redundant call to the ReleaseDC function was detected. This error may be caused by incorrect window or device-context parameters. This error message is sent by the USER module.

ReleaseDC: hwnd not same as for GetDC  
The device context or window parameter to the ReleaseDC function is incorrect. The window or device context is not the same as was passed to (or returned from) the GetDC function. This error message is sent by the USER module.

ReleaseDC: Passed DC not a window DC  
The device context supplied to the ReleaseDC function was not obtained by using the GetDC function. This error is often caused by attempting to release printer or memory device contexts. This error message is sent by the USER module.

Resources N% - this tests your error handling code  
The debugging kernel allows applications to load even with very low system resources. This allows the application developer to test how their code handles allocation failure. This warning demonstrates the different behaviour of the debugging kernel and retail kernel. This warning message is sent by the KERNEL module.

Seek failed.  
A disk or link error occurred. This error message is sent by the KERNEL module.

Segment N of module-name must be preload  
Certain segments of an application or DLL must be preloaded. If an application sets the switch incorrectly, the system corrects the setting. However, the application will load faster if it sets the switch properly. The following segments must be preloaded: DATA segments, FIXED CODE segments, and CODE segments that are MOVEABLE but not DISCARDABLE. This warning message is sent by the KERNEL module.

Segment N of module-name was discardable under Win 3.0  
In Windows 3.1, the DISCARDABLE bit is not necessarily set for DLL segments that are marked MOVEABLE. In Windows 3.0, DLL segments marked MOVEABLE were also made DISCARDABLE. In general, if a DLL segment is MOVEABLE, it can also be DISCARDABLE. This warning message is sent by the KERNEL module.

SetClassWord: Invalid class brush  
The SetClassWord function was called with an invalid brush handle. This error message is sent by the USER module.

SetSysModalWindow failed: another app's window already sys modal

The SetSysModalWindow function failed because another application was already displaying a system modal window. This warning message is sent by the USER module.

SetWindowLong/SetClassLong of NULL window procedure  
An attempt was made to subclass a window by using a NULL or invalid window procedure address. This warning message is sent by the USER module.

SetWindowPos: Invalid hwndInsertAfter  
The hwndInsertAfter parameter to the SetWindowPos or DeferWindowPos function is invalid. This error may occur because a window was destroyed between the call to the DeferWindowPos function and the call to the EndDeferWindowPos function. This error message is sent by the USER module.

SetWindowPos: Invalid window handle  
An invalid window handle was passed to the SetWindowPos or DeferWindowPos function. This error may occur because a window was destroyed between the call to the DeferWindowPos function and the call to the EndDeferWindowPos function. This error message is sent by the USER module.

SetWindowPos: WS\_EX\_TOPMOST window positioned incorrectly  
The WS\_EX\_TOPMOST windows have been corrupted. This internal error message is sent by the USER module.

SetWindowsHook called to unhook: use UnhookWindowsHook  
The SetWindowsHook function was called with the window hook address returned from a previous call to SetWindowsHook. This can result in unhooking more than one hook from the hook chain. The UnhookWindowsHook function should always be used to unhook hooks. This warning message is sent by the USER module.

SetWindowsHook: HookProc must be in a DLL  
Hook functions with system scope that are installed by using the SetWindowsHookEx function or hook functions installed by using the SetWindowsHook function (other than WH\_MSGFILTER hooks) must be defined in a DLL. Task hooks and WH\_MSGFILTER hooks can be defined in a standard executable file. This warning message is sent by the USER module.

Starting Code Segment of module-name must be preload  
Certain segments of an application or DLL must be preloaded. If an application sets the switch incorrectly, the system corrects the setting. However, the application will load faster if it sets the switch properly. The following segments must be preloaded: DATA segments, FIXED CODE segments, and CODE segments that are MOVEABLE but not DISCARDABLE. This warning message is sent by the KERNEL module.

System message box already up  
An internal error occurred that caused a system modal message box to be displayed when there was already one displayed. This error message is sent by the USER module.

Too many windows positioned with tasks locked  
An internal buffer has overflowed. This internal fatal error message is sent by the USER module.

TrueType font width mismatch  
There is a possible error in the selected TrueType font. This warning message is sent by the GDI module.

Unable to load filename (number)  
A startup error occurred. This error message is sent by the KERNEL module.

Unallocated extra window/class word index used  
An invalid index was used in a call to the GetWindowWord, SetWindowWord, GetClassWord, or SetClassWord function. This warning can occur when these functions are called with windows created by other applications or other parts of the application. An application should not call these functions unless it is guaranteed that the window class supports the extra window words. This warning also can occur if insufficient window or class words are allocated when the window class is registered. This warning can occur in the retail version of Windows. The LogError constant is ERR\_STRUCEXTRA. This warning message is sent by the USER module.

Unknown fixup N  
A disk or link error occurred. This error message is sent by the KERNEL module.

UnlinkWin386Block: releasing 0xNNNNNN bytes  
Memory is being returned to a DPMS server (Win386 or DOSX). This trace message is sent by the KERNEL module to help applications optimize memory management.

UnregisterClass failed: called from wrong app  
The UnregisterClass function must be called from same application that called the RegisterClass function. This error message is sent by the USER module.

UnregisterClass failed: class doesn't exist  
The UnregisterClass function failed because the specified class does not exist. This error message is sent by the USER module.



UnregisterClass failed: class windows still exist

The UnregisterClass function failed because windows for the specified window class have not been destroyed. This error message is sent by the USER module.

Use of DC after ReleaseDC or EndPaint

Drawing occurred in a device context after it was released with the ReleaseDC or EndPaint function. This is a serious error; it can cause drawing to occur in other application windows. This error is often caused by an application neglecting to set global HDC variables to NULL after releasing the device context. This error message is sent by the USER module.

USER: Menu not destroyed: 0x1234

The specified menu handle was not destroyed by an application before terminating. This warning message is sent by the USER module.

USER: Window not destroyed: 0x1234

The specified window handle was not destroyed by an application before terminating. This warning message is sent by the USER module.

Warning: Yield() during application termination

An application yielded during application termination. This message can help track down timing problems that occur while the application is terminating and are otherwise difficult to debug. This warning message is sent by the USER module.

Window class freed with existent class windows!

An internal error occurred when the system was destroying the windows created by an application while terminating the application. This error message is sent by the USER module.

Window class reference count overflow

The internal system data structures may have been damaged. This error could be caused by writing through an uninitialized pointer or performing other kinds of incorrect memory handling. This error message is sent by the USER module.

Window class reference count underflow

The internal system data structures may have been damaged. This error could be caused by writing through an uninitialized pointer or performing other kinds of incorrect memory handling. This error message is sent by the USER module.

Window destroyed itself during WM\_DESTROY processing

The DestroyWindow function was called a second time while the window was processing the WM\_DESTROY message. This error message is sent by the USER module.

Window destroyed unexpectedly by callback

A window was destroyed unexpectedly while it was processing a message. This warning message is sent by the USER module.

Windows will delete class brushes

A class brush is invalid when the application exits. When an application is terminating, all classes it registered are destroyed. The class brush handles are also deleted. If an application deletes its class brush before terminating, this message will be generated. This warning message is sent by the USER module.

WM\_NCACTIVATE FALSE return ignored during WM\_MDIACTIVATE

In Windows version 3.0, the return value of the WM\_NCACTIVATE message is ignored when the message is sent to a MDI child window. In Windows 3.1, the activation is prevented (just as with top-level windows) if FALSE is returned. This error message is provided for applications that depend on compatibility with Windows 3.0 to indicate that a FALSE return value is ignored in an application for Windows 3.0. This error message is sent by the USER module.

WM\_NCACTIVATE FALSE return ignored: activating sys modal window

Returning FALSE after processing the WM\_NCACTIVATE message does not prevent a window from being deactivated if a system modal window is activated. This warning message is sent by the USER module.

WS\_CLIPCHILDREN overridden by CS\_PARENTDC

In Windows 3.1, the WS\_CLIPCHILDREN style works correctly when a window is created with the CS\_PARENTDC style. In Windows 3.0, the WS\_CLIPCHILDREN style has no effect in this situation. This warning message is sent by the USER module.

WS\_CLIPSIBLINGS overridden by CS\_PARENTDC

In Windows 3.1, the WS\_CLIPSIBLINGS style works correctly when a window is created with the CS\_PARENTDC style. In Windows 3.0, the WS\_CLIPSIBLINGS style has no effect in this situation. This warning message is sent by the USER module.

wsprintf: Invalid char sequence follows '%'

An improper format string was passed to the wsprintf function. This warning message is sent by the USER module.

Zero import module.

A disk or link error occurred. This error message is sent by the KERNEL module.

## Common Programming Errors

The following list describes programming errors that sometimes appear in Windows applications:

- Passing invalid parameters.
- Accessing nonexistent window words. (In Windows 3.0, a call to the SetWindowWord or SetWindowLong function past the end of the allocated window words, as defined by the RegisterClass function, would damage internal window-management structures.)
- Using handles after they have been deleted or destroyed.
- Using a device context after it has been released.
- Deleting GDI objects before they are selected out of a device context.
- Neglecting to delete GDI or USER objects when an application terminates.
- Writing past the end of an allocated memory block.
- Reading or writing using a memory pointer after it has been freed.
- Neglecting to export window procedures and other callback functions.
- Neglecting to use the MakeProcInstance function with dialog procedures and other callback functions.

Many of these programming errors can cause unrecoverable application errors in Windows version 3.0. The debugging system can help you locate these types of problems.

## Compatibility Issues

Although every effort has been made to ensure that the many enhancements and improvements to the Windows operating system, version 3.1, are compatible with Windows 3.0 applications, some enhancements may affect application operation. This is especially true if an application uses features in an undocumented fashion or relies on invalid assumptions about the behavior of Windows.

The following Help topics discuss categories of compatibility issues:

Window Management

TrueType

Undocumented Windows 3.0 Features

## Window Management Compatibility Issues

The window management (USER module) enhancements may affect Windows 3.0 applications. To test this, you need to perform as many operations as possible that cause your application windows to be moved, sized, scrolled, and repainted. The following sections identify a few basic methods to try, but you should try as many other methods as possible.

In some cases, Windows 3.1 ensures compatibility with existing Windows 3.0 applications by supporting both the Windows 3.0 and the new Windows 3.1 implementations. If an application's Windows version as set by Microsoft Windows Resource Compiler (RC) is 3.0, Windows 3.1 carries out the Windows 3.0 implementation, meaning that the Windows 3.1 enhancement has no impact on an existing Windows 3.0 application. However, if a Windows 3.0 application's Windows version is changed to 3.1 without corresponding changes to the application code, the application may encounter problems whenever Windows 3.1 carries out the 3.1 implementation.

### Moving and Sizing

#### MoveWindow

A call to the MoveWindow function is equivalent to a call to SetWindowPos with the SWP\_NOZORDER and SWP\_NOACTIVATE flags set. If the MoveWindow fRedraw parameter is FALSE, the SWP\_NOREDRAW flag is also set. For Windows 3.0 applications, when MoveWindow is called for a top-level window with fRedraw set to FALSE, Windows calls SetWindowPos without setting the SWP\_NOREDRAW flag and then calls the ValidateRect function to prevent the client area from being repainted. However, WM\_NCPAINT and WM\_ERASEBKGD messages will have been sent, even though fRedraw was FALSE. For Windows 3.1 applications, MoveWindow no longer sends these messages in this special case.

For Windows 3.0 applications, Windows always completely redraws a window's frame when the window is moved or sized. For Windows 3.1 applications, Windows no longer completely redraws a window's frame in all cases. For example, the following code sequence does not redraw the window border:

```
MoveWindow(hwnd, ..., FALSE);
.
.
.
.
.
InvalidateRect(hwnd, NULL, TRUE);
```

#### SetWindowPos

For Windows 3.0 applications, the SetWindowPos function assumes that SWP\_NOMOVE and SWP\_NOSIZE are set if SWP\_HIDEWINDOW or SWP\_SHOWWINDOW is set. This means it is not possible in an atomic operation both to hide or show a window and to change its size or position. For Windows 3.1 applications, this limitation does not exist.

For Windows 3.0 applications, when the window is already visible, a call to SetWindowPos with the SWP\_SHOWWINDOW flag set always causes the entire window to be redrawn. This also affects the operation of the ShowWindow function. For Windows 3.1 applications, when the window is already visible, a call to SetWindowPos with SWP\_SHOWWINDOW sets does not cause the window to be redrawn (unless another area must be updated as a result of a size, move, or z-order operation specified in addition to SWP\_SHOWWINDOW).

#### WM\_NCCALCSIZE

In Windows 3.0, the WM\_NCCALCSIZE message is always sent to a window whenever the window is moved or sized. For Windows 3.1, the WM\_NCCALCSIZE message is sent only if the size of the window actually changes.

### Painting

Window management has been substantially optimized to avoid unnecessary redrawing and flashing. Applications that depend in subtle ways on when (and if) WM\_NCPAINT, WM\_ERASEBKGD, and WM\_PAINT messages are sent may have incompatibilities. Windows 3.0 frequently sent these messages redundantly to windows and sometimes sent them to windows that were not visible. One of the visual results of the Windows 3.1 optimizations is that a window's nonclient area is not always completely repainted when a window is sized or moved. Some attempt has been made to ensure compatibility, but there are some differences that cannot be backward-compatible and still achieve the significant performance and visual advantages.

#### BeginPaint and GetDC

For Windows 3.0 applications, if the BeginPaint function is called on a window that has a class icon, the function returns a window device context (DC); in contrast, the GetDC function returns a client DC with no visible region. For Windows 3.1 applications, BeginPaint and GetDC both return a client DC with no visible region.

#### BS\_USERBUTTON Style

The BS\_USERBUTTON style is not valid for Windows 3.1.

#### CS\_PARENTDC Class Style

For Windows 3.0, a window with the CS\_PARENTDC class style whose parent window does not have the WS\_CLIPCHILDREN style receives the device context of the parent window, even when the child window has the WS\_CLIPSIBLINGS or WS\_CLIPCHILDREN style. For Windows 3.1, a window with the CS\_PARENTDC class style does not receive the parent device context if the window style is either WS\_CLIPSIBLINGS or WS\_CLIPCHILDREN. Windows 3.1 consistently favors the window-style specifications over the class style.

#### GetUpdateRect

For Windows 3.0 applications, calling the GetUpdateRect function for a window that has the CS\_OWNDC class style and a mapping mode other than MM\_TEXT sometimes retrieves the update rectangle in device coordinates instead of logical coordinates. For Windows 3.1 applications, the result is always in logical coordinates for this class style and mapping mode.

#### InvalidateRect and InvalidateRgn

For Windows 3.0 applications, when the InvalidateRect or InvalidateRgn function is called with the lprc parameter set to NULL to invalidate the entire window, all child windows are also completely invalidated--regardless of whether the child window is outside the parent's client area (that is, invisible). This results in WM\_PAINT messages being sent to windows that don't require them. For Windows 3.1 applications, only windows that are actually visible within a parent's client area receive update regions and therefore receive WM\_PAINT messages.

#### InvalidateRect and RedrawWindow

A minimized Windows 3.0 application could call the InvalidateRect function to invalidate its icon. For a Windows 3.1 application to invalidate its icon, it must call the RedrawWindow function and specify RDW\_FRAME for the fuRedraw parameter.

#### Multicolumn List Boxes

A multicolumn list box in Windows 3.0 always received two paint messages when being created. For Windows 3.1, a multicolumn list box receives only one paint message.

#### UpdateWindow

In Windows 3.0, the various controls call the UpdateWindow function at inappropriate times, such as when receiving a WM\_SETFOCUS message and at other times when changing the internal state. Some controls may not be redrawn properly if they are moved or hidden before they are able to process a WM\_PAINT message. In Windows 3.1, the controls do not call UpdateWindow as often, resulting in faster window repainting and improved appearance.

#### WM\_DRAWITEM

For list boxes in Windows 3.0, the wParam parameter of the WM\_DRAWITEM message is always zero. In Windows 3.1, the wParam parameter specifies the identifier of the control that sent the message.

#### WM\_ERASEBKGD

For Windows 3.0 applications, if an application responds with FALSE to a WM\_ERASEBKGD message sent during any operation other than BeginPaint (such as SetWindowPos), another WM\_ERASEBKGD message is sent when the application calls BeginPaint. For Windows 3.1 applications, if an application responds with FALSE, no second WM\_ERASEBKGD message is sent but BeginPaint sets the fErase member of the PAINTSTRUCT structure to TRUE.

For Windows 3.0 applications, calls to the InvalidateRect function with the fErase parameter equal to FALSE always prevented the window from receiving a WM\_ERASEBKGD message, even if the message was already pending before the call to InvalidateRect was made. For Windows 3.1, pending WM\_ERASEBKGD messages are received by the application.

#### WM\_SETREDRAW

For Windows 3.0 applications, sending the WM\_SETREDRAW message with the wParam parameter set to FALSE to a window that has an update area does not validate the window. The update area is still present after a WM\_SETREDRAW message with wParam set to TRUE. For Windows 3.1 applications, sending WM\_SETREDRAW with wParam set to FALSE does validate the window completely to ensure that the window does not receive any WM\_PAINT messages while it is invisible. This does not apply to edit controls, list boxes, and combo boxes, because their WM\_SETREDRAW messages are handled differently.

#### WM\_SETVISIBLE

For Windows 3.0 applications, Windows sends a WM\_SETVISIBLE message immediately after sending the WM\_SHOWWINDOW message. For Windows 3.1 applications, Windows does not send the WM\_SETVISIBLE message-- WM\_SETVISIBLE is obsolete for Windows 3.1.

### Scrolling

For Windows 3.0 applications, the ScrollWindow function has a number of bugs associated with scrolling a window that had any invalid area. Frequently, the update region resulting from the scrolling operation is not properly calculated. For Windows 3.1 applications, ScrollWindow calculates the update region correctly.

### Multiple Document Interface (MDI)

Multiple document interface (MDI) is completely compatible with Windows 3.0 applications. For Windows 3.1 applications, MDI has been enhanced. In particular, specifying the low-order style bit (MDIS\_ALLCHILDSTYLES) when creating an MDICLIENT window enables the new Windows 3.1 MDI capabilities for that window. This gives applications control over all MDI child window styles and allows for hidden windows.

### Windows Hooks

#### Hook Chain

In Windows 3.0, an application or dynamic-link library (DLL) that installs a hook is responsible for maintaining the hook chain. In Windows 3.1, Windows maintains the hook chain. Consequently, there are subtle changes in the interface that may affect Windows 3.0 applications. Furthermore, Windows 3.1 no longer allows applications and DLLs to enumerate all the functions in a hook chain. In particular, Windows 3.1 no longer supports the HC\_GETLPLPFN, HC\_LPLPFNNEXT, and HC\_LPFNNEXT values for the DefHookProc function.

#### Negative Hook Values

In Windows 3.0, Windows passes a negative hook value to a hook function when unhooking a hook. This negative value is for Windows internal use only. In Windows 3.1, Windows does not pass a negative hook value to a hook function; it uses another method to unhook a hook.

#### SetWindowsHook

In Windows 3.0, the SetWindowsHook function returns a pointer to the next hook function. In Windows 3.1, SetWindowsHook does not return a pointer; instead, it returns a 32-bit value that identifies the next hook function. An application that attempts to call the hook function by using the return value from SetWindowsHook as a function address causes a general protection (GP) fault.

In Windows 3.0, an application can unhook a hook function by passing the address of the next hook function to the SetWindowsHook function. In Windows 3.1, passing the address of the next hook function causes a system debugging error (RIP) in the Windows 3.1 debugging version.

#### SetWindowsHookEx, UnhookWindowsHookEx, and CallNextHookEx

In Windows 3.0, three hook functions are available: SetWindowsHook, UnhookWindowsHook, and DefHookProc. In Windows 3.1, these functions are replaced with three more powerful functions: SetWindowsHookEx, UnhookWindowsHookEx, and CallNextHookEx. Windows 3.1 applications should use the new functions. The old functions are still supported for backward compatibility.

### Parameter Validation

Windows strictly checks parameters passed to its functions before using them. For Windows 3.0 applications, there are many validation errors that Windows works around and lets the function or application continue to function. For Windows 3.1 applications, many of these errors cause the functions to fail and it is up to you to ensure that structures and parameters are passed correctly.

For example, in Windows 3.0, if an application passes NULL as the hInstance parameter to CreateWindowEx, Windows maps the handle to the stack segment. In Windows 3.1, the function returns an error value.

### Undeleted Object Notifications

In Windows 3.1, any GDI or USER object left allocated when an application terminates results in a warning to the debug terminal. Windows 3.1 does not automatically free these objects--your application must free them. These warnings usually imply a memory leakage, in which case running and terminating an offending application eventually uses all available memory.

Sometimes an object is intended to last longer than the application or DLL that created it. This occurs frequently in shared DLLs that share GDI objects such as bitmaps and brushes among its many clients. In such cases, the warnings at the debug terminal can be ignored.

## Menu Implementation

### SendMessage and PostMessage

For Windows 3.0 applications, Windows uses the SendMessage function to send a WM\_COMMAND message. For Windows 3.1 applications, Windows uses the PostMessage function to send the message, preventing stack overflow when the application is working with pop-up menus.

### TrackPopupMenu

Menu management has been enhanced for Windows 3.1. In particular, the TrackPopupMenu function now allows additional parameters, and Windows now stores application menu data in a separate heap, expanding the number of windows that can exist.

## RegisterClass

In Windows 3.0, Windows fails to properly free the window-class background brush when deleting the class. In Windows 3.1, Windows frees the brush when either a Windows 3.0 or 3.1 application terminates.

## Topmost Window

A new window attribute allows a window to be placed on top of all other windows, even when the owning application is not active. If multiple applications have topmost windows, the topmost windows will have the same order as their owning applications.

Also, a topmost window, its owners, and all the windows it owns will stay together as windows are moved around. This means that if you bring an owned dialog window to the top, its owner will also be brought forward so that it stays immediately below the dialog box.

An application that depends on being able to have a window of another application between its main window and a dialog box may encounter problems. For example, a setup program that starts Windows Notepad and then brings up a dialog box causes Notepad to be positioned behind the dialog box owner. The solution in this case is to create the dialog box without an owner (a window cannot be owned by a window of another application).

Any application that relies on having an unobstructed client area when the application is active may encounter problems, because it is not possible to guarantee that the active window is on top. This means the active window may not have a rectangular clipping region, because a topmost window may be on top of it. Calling the BitBlt function with a window or screen DC as the source (which is not recommended in any case) may copy bits belonging to the topmost window.



## TrueType Compatibility Issues

### TrueType Compatibility Issues

Although Windows 3.1 includes support that seamlessly integrates TrueType fonts into existing applications, problems with fonts can occur for Windows 3.0 applications that assume bitmap fonts are always available, that Helv and Tms Rmn font are always available, and that font sizes are limited. Be sure to thoroughly check fonts in your application, including files and dialog boxes. Also, because TrueType provides more fonts in more styles, Windows 3.1 may consume both printer and global memory faster than Windows 3.0. You should check your applications with systems and printers that have limited memory.

### Helv and Tms Rmn Fonts

Helv and Tms Rmn fonts are no longer available. The fonts that replace these are MS Sans Serif and MS Serif, respectively. To support Windows 3.0 applications that use the Helv and Tms Rmn fonts, the [FontSubstitutes] section in the WIN.INI file maps Helv to MS Sans Serif and Tms Rmn to MS Serif by default. It also maps Times® to the Times New Roman TrueType font and Helvetica® to the Arial TrueType font.

Applications that search explicitly for "Helv" or "Tms Rmn" may encounter difficulties when these fonts are not found.

### Font Enumeration

Applications should test to ensure that TrueType fonts are enumerated correctly. Applications should also ensure that they encounter no unexpected font mapping. (When a TrueType font substitutes for another font, line spacing, paragraph breaks, or page breaks could change.)

Windows 3.0 applications sometimes create multiple instances of a single font or font family. In particular, some applications use different handling for fonts that are enumerated by a nonraster printer than they use for fonts enumerated for the screen, even if these fonts have the same names. With TrueType fonts, fonts with the same name are identical, regardless of the output device. Some Windows 3.0 applications assume that scalable fonts can not be available on nonscaling devices. In such cases, the applications intentionally enumerate a single size for every TrueType font even though other sizes are available. Furthermore, some applications assume that bold, italic, and bold italic are always simulated from regular fonts. This is not true with TrueType fonts.

An application can create multiple instances of the same font.

Windows continues to support and is fully backward-compatible with Adobe Type Manager (ATM), Facelift, and Intellifont for Windows. Applications using these font technologies should encounter no problems.

### TrueType Only

Windows 3.0 applications may behave unexpectedly if the user has used Control Panel to check the Show Only TrueType Fonts in Applications check box. An application may fail to locate any fonts if only TrueType fonts are present.

### Font Sizes

TrueType supports a wide variety of sizes for all TrueType fonts. In Windows 3.1, if an application requests a very small or very large font, it usually gets the requested size.

An application that checks for the smallest or largest font by setting the nHeight parameter in the CreateFont function to an extreme value will not get the expected results.

### Font Substitutions

The [FontSubstitutes] section may cause the GetTextFace function to return a typeface name that is not enumerated by the EnumFontFamilies function. This ensures that an application gets the typeface name it requests. For example, an application that requests Helv (and expects Helv) gets a typeface named Helv.

An application that expects matching facenames from EnumFontFamilies and GetTextFace may encounter mismatches.

### ABC Spacing

ABC-spaced fonts can lead to misplaced cursors, highlights that do not encompass all the text on a line,

pieces of characters left behind after screen updates, and unexpected clipping of fonts on printers (when a character goes outside the printable area).

#### Third-Party Font Manager Problems

Be sure to try your application with ATM, Facelift, or Intellifont for Windows fonts installed. Do not install more than one of these font managers at a time. Skip this test if your application does not work with these font managers under Windows 3.0.

#### Mixing Device, Bitmap, and TrueType Fonts

In Windows 3.1, some fonts, such as the Symbol font, may be supported by a TrueType font, a GDI bitmap font, and a device-specific font. Applications can get unexpected results if they specify the name of the font without specifying the font technology; for example, the Symbol bitmap font could be mixed with the Symbol TrueType font in a print job.

Printing may mix device, bitmap, and TrueType fonts, causing unacceptable output.

#### Desktop Publishing and International Characters

Windows 3.1 includes 22 new international and desktop publishing characters. Unfortunately, these new characters appear only in TrueType fonts; the bitmap fonts do not have them.

Changing to a bitmap font causes the new characters to be displayed as the default character for the current font. Some applications may perform their own remapping in the ASCII character range 128 through 159.

Note: The desktop publishing characters are not available to dialog boxes that use bitmap fonts exclusively (such as the Find and Replace dialog boxes).

#### Text Rotation

Although Windows 3.0 can rotate vector and device fonts, under certain mapping modes it rotates these fonts differently. For compatibility, Windows 3.1 also rotates fonts differently. However, an application can override this default behavior and direct Windows 3.1 to use the same convention to rotate all fonts by setting the CLIP\_LH\_ANGLES bit in the lfClipPrecision member of the LOGFONT structure. When this bit is set, Windows 3.1 rotates all fonts using the same rules used by Windows 3.0 to rotate device fonts.

#### Other TrueType Considerations

Some applications do not request point sizes correctly. For bitmap fonts, the results are acceptable because only these fonts have a limited range of sizes available. For TrueType fonts, output can be unacceptable because any size requested is available.

Windows 3.0 applications sometimes set the tmAveCharWidth member of the TEXTMETRIC structure to request a specific font. With Windows 3.1, the widths of TrueType characters are changed to match the requested width.

With TrueType, Windows now adds at least 13 fonts to the default list. Some applications may fail because they do not have test cases that account for the additional fonts.

#### Printing

You can evaluate the effects of most of the printing changes by printing documents from your application in Windows 3.0 and Windows 3.1 and comparing the output. Although you should test as many printers as possible, you must test the following four printer types:

- PostScript
- LaserJet II
- LaserJet III
- Dot matrix

Some printers have been renamed. This could cause problems in your application if your application or documents make specific references to a certain printer. Note that renaming the printer does not affect soft fonts.

#### Other TrueType Enhancements

The font enhancements implemented by TrueType do not cause problems for applications unless the applications depend upon internal structures or internal operations. TrueType enhancements can cause problems in the following situations:

- When an application depends upon internal data structures, which may have changed.
- When an application relies on real mode, which is no longer supported.

- When an application builds its own selectors, KERNEL now runs at ring 3 (instead of ring 1, as in Windows 3.0) to prevent ill-behaved applications from relying on the relationship between handles and selectors.

## Undocumented Windows 3.0 Features

### Undocumented Windows 3.0 Features

Windows 3.0 applications that call undocumented Windows 3.0 functions or structures may fail when run with Windows 3.1. Most of the undocumented functions are internal USER, GDI, and KERNEL functions that Windows 3.0 must export to support movable data segments in real mode. Windows 3.1 supports only protected mode. Therefore, Windows 3.1 does not export these internal functions, and applications that attempt to link with them will fail.

The undocumented structures are internal structures used by Windows to store information about Windows objects, such as windows and device contexts. To support parameter validation, Windows 3.1 has changed the number and meaning of the members in these structures. Windows 3.0 applications that directly access these structures will eventually fail.

To determine whether your application will fail, run it with the debugging version of Windows 3.1. The debugging version displays an error message if the application contains undocumented functions that are no longer supported. Windows closes the application when the unsupported function is actually called.

## Creating Windows Applications

This topic explains what elements are needed to build applications for the Microsoft Windows operating system versions 3.0 and 3.1. It also provides guidelines for writing robust applications and for debugging applications.

### Writing Compatible Windows Applications

The Microsoft Windows 3.1 Software Development Kit (SDK) allows you to create applications for either Windows 3.0 or 3.1. If you write your application carefully, you can create a single application that is compatible with Windows 3.0 but also takes advantage of newer features when running with Windows 3.1.

### Windows 3.1 Applications

The Windows 3.1 SDK tools, header files, and libraries create Windows 3.1 applications by default. No special procedures are required to create executable files that run with Windows 3.1. If you create Windows Help files for your applications, use Microsoft Help Compiler version 3.1 (HC31.EXE) to compile your files so that they have access to the latest features of Windows Help.

Applications that call Windows 3.1 functions depend on Windows 3.1 and cannot be run with Windows 3.0.

### Windows 3.0 Applications

You can use the Windows 3.1 SDK to create a Windows 3.0 application by following these steps:

- 1 Set the WINVER define variable to 0x300 to enable the WINDOWS.H file for Windows 3.0 compilation. Place the following statement immediately before the include statement for the header file:

```
#define WINVER 0x300
```

- 2 Link your application object files with the LIBW.LIB library provided with the Windows 3.1 SDK. Except for the functions that are new to Windows 3.1, all functions defined in this import library are compatible with Windows 3.0.
- 3 Mark your application as a Windows 3.0-only executable by using the /30 option with Windows Resource Compiler (RC). The /30 option cannot be used with the /r option.
- 4 If you create Windows Help files for your application, use Help Compiler version 3.0 (HC30.EXE) to compile your files.

By default, Resource Compiler marks applications for 3.1, so it is important to use the /30 option mentioned in the preceding steps.

All Windows 3.0 applications can use Windows extensions, such as common dialog boxes and object linking and embedding. If you use these features, you must ship the corresponding dynamic-link libraries (DLLs) and related files with your application. They should be installed along with the application.

### Combined Windows 3.0 and 3.1 Applications

You can create Windows applications that run with Windows 3.0 but also take advantage of newer features when running with Windows 3.1. Such applications consist primarily of Windows 3.0 function calls but conditionally link to and use Windows 3.1 functions.

To build a combined application, mark your application as a Windows 3.0 only executable by using the /30 option with Resource Compiler, but do not set the WINVER define variable to 0x300. You must use the GetVersion function to determine the version of Windows that is running before using any Windows 3.1 functions.

The following example demonstrates how to set a flag if the current system is Windows 3.1:

```
extern BOOL fWin31;

UINT version;

fWin31 = FALSE;

version = LOWORD(GetVersion());
if (((LOBYTE(version) << 8) | HIBYTE(version)) >= 0x030a) {
    fWin31 = TRUE;
}
```

```
}
```

For information about interpreting the return value of the GetVersion function, see the Microsoft Windows Programmer's Reference, Volume 2.

Your application can call Windows 3.1 functions directly as long as you link it with the 3.1 version of LIBW.LIB. (It is not necessary to call the GetProcAddress function.) However, you must ensure that Windows 3.1 functions are not called when your application is running with Windows 3.0. The following example demonstrates how this can be done, using the `fWin31` flag that was set in the preceding example:

```
extern BOOL fWin31;

if (fWin31) {
    ScrollWindowEx(hwnd, ...); /* new for Windows 3.1 */
}
else {
    ScrollWindow(hwnd, ...); /* Windows 3.0 function */
}
```

If you create Windows Help files for your application, either use Help Compiler version 3.0 (HC30.EXE) to compile your files, or create a help file for Windows 3.1 and release your help file with the redistributable Windows 3.1 versions of the WINHELP.EXE and WINHELP.HLP files.

If you run a combined application using the debugging version of Windows 3.0, a call to an undefined function causes a warning. The application, however, continues to load and run successfully, as long as the function is not actually called.

### Creating Robust Applications

Windows 3.1 includes a number of features and enhancements designed to make running Windows applications much more reliable. Efforts to make Windows 3.1 more reliable have focused primarily on three areas:

- Improving how the system handles errors if and when they occur.
- Avoiding errors in system code by ensuring the validity of all handles, pointers, structures, indices, and flags passed to the system.
- Providing better diagnostics, tools, and header files for finding and fixing bugs more efficiently during development.

The two key components improving reliability are the parameter validation built into the Windows operating system and the STRICT type-checking of the WINDOWS.H file. Also useful are the new features of WINDOWSEX.H, which include macros, message crackers, and control functions.

### Parameter Validation

Windows 3.1 contains code to validate parameters passed to Windows functions and messages. These features are included in both the retail and debugging versions of the system. The debugging version of the system includes some additional features and parameter checking that is not included in the retail product.

### Invalid Parameter Error Messages

The system validates handles, pointers, structures, indices, and flags. In most cases, an invalid parameter causes a function to return an error value. In other cases, such as when a flag is invalid, the function executes as usual, but an appropriate warning message is displayed.

When Windows encounters an invalid parameter error, it displays the message on your debugging terminal or window. The message has the following form:

```
err AppName function:address: message:parameter-value
```

Following are the message parameters:

AppName	Identifies the application or DLL that caused the error.
function	Identifies the number of the function that was passed the invalid parameter.
address	Identifies the address of the function that was passed the invalid parameter.
message	Specifies the string identifying the error.
parameter-value	Specifies the value of the invalid parameter.

For example, a message could have the following form:

```
err FONTSAMP 011F:056A: Invalid local handle: 1D50
```

If the address is not near the address of a Windows function you recognize, the window message parameter is probably invalid. Functions that take messages, such as SendMessage, DispatchMessage and SendDlgItemMessage, show an address within the message validation code. Parameter values for invalid parameters begin with the PV prefix (for example, PV\_WM\_COMMAND).

By default, invalid parameter messages display a stack trace and an "Abort, Break, or Ignore?" prompt. You can change the default by setting options in the System Debug Options box of the Systems Debugging Log Application (DBWIN.EXE). This dialog box is displayed when you choose the Settings command on the Options menu.

You can also log invalid parameter errors by using Dr. Watson, just as you would log general-protection (GP) faults by using Dr. Watson. By default, this feature is turned off.

#### Buffer Overflow Errors

A common application error is to allocate too little space for a buffer that is passed to and filled by Windows. These errors are especially difficult to track if the buffers are allocated on the stack. Windows can help you find these errors by filling buffers before information is copied into them. If the operation overflows the buffer, Windows detects and reports the error.

By default, this feature is disabled. You can enable the feature by choosing the Settings command on the Options menu of DBWIN.EXE and then selecting the Fill Buffers check box. When you select this check box, Windows displays a stack trace and an "Abort, Break, or Ignore?" prompt with some warning messages.

This feature is available in the Windows debugging version only.

#### Interpreting Invalid Parameters

Possible reasons for getting invalid parameter errors follow.

##### Invalid Handles

A handle is invalid under the following circumstances:

- Using NULL or -1 when it is not allowed
- Reusing a destroyed or deleted handle
- Using an uninitialized stack variable
- Using a device context handle created by the CreateIC or CreateMetaFile function in a function that does not allow the handle
- Passing one type of handle in place of another, such as passing a device-context handle in place of an window handle

##### Invalid Pointers

A pointer is invalid under the following circumstances:

- Using a NULL pointer when it is not allowed
- Pointing to a buffer that is too small
- Using a function pointer without properly exporting it or properly creating a procedure-instance address
- Pointing to a string that does not have a null-terminating character
- Pointing to a structure that contains an invalid member (for example, if you call the RegisterClass function with a invalid window procedure in the CREATESTRUCT structure, Windows reports an invalid pointer)
- Using an uninitialized stack variable
- Passing a read-only pointer when a read-write pointer is required

##### Invalid Flags or Value

A flag or value is invalid under the following circumstances:

- Passing meaningless flags
- Passing an out-of-range index
- Using a value that is otherwise illegal

You can use pointer-validation functions (such as IsBadCodePtr) to help you check for and debug your application's use of pointers.

#### Strict Type-Checking

The Windows 3.1 header file (WINDOWS.H) includes various features for detecting problems when compiling an application. These features, which are provided by the STRICT option, make application development faster and easier.

You define `STRICT` before the include statement for the header file. `STRICT` causes the various types and function prototypes in `WINDOWS.H` to be declared with very strict type-checking. For example, once `STRICT` is defined, it is impossible to pass a window handle to a function that requires a device-context handle without generating a compiler error.

### Features of the `STRICT` Option

Specific features provided by the `STRICT` option include:

- Strict handle type-checking
- Proper declaration of certain parameter and return value types
- Fully prototyped type definitions for callback function types
- Proper declaration of polymorphic parameters and return values (for example, `wParam` and `IParam` message parameters)
- Proper use of the `const` keyword for pointer parameters and structure members when the pointer is read-only

Type declarations for many of the Windows functions and callback functions have changed. Nonetheless, unless you define `STRICT`, the new declarations for Windows 3.1 are fully compatible with the old declarations for Windows 3.0. `WINDOWS.H` for Windows 3.1 can, therefore, be used to compile Windows 3.0 applications without modifications.

### Compiling with the `STRICT` Option

In general, the `STRICT` option is most useful with newly developed code or with code that is being maintained or changed regularly. Code that has already been written and tested, and is not changing very much over time, will generally not benefit as much from `STRICT`. If you find that stable code generates lots of run-time parameter validation errors when run with Windows 3.1, you will find `STRICT` very valuable as you go through the code to clean up those errors.

The following procedures will ensure that your application conforms to `STRICT` type-checking:

- Use new handle and parameter types. In particular, replace `HANDLE` with appropriate handle types and use `WPARAM` and `LPARAM` with all message parameters.
- Use new return type and parameter types for windows and dialog box procedures and callback functions.
- Declare all your functions with full prototypes. Place these prototypes in a include file and include it with each source file.
- Cast function pointers to the proper type rather than to `FARPROC` type. This is especially important with the `MakeProcInstance` function.
- Take special care with `HMODULE` and `HINSTANCE` types. There are a few Windows functions that return or accept only `HMODULE` types.
- Use the `MAKELPARAM` macro instead of the `MAKELONG` macro when building `LPARAM` parameters out of two words. Also, use the `MAKELRESULT` macro instead of `MAKELONG` when building `LRESULT` return values.
- Cast the handle or near pointer to a `WORD` type in order to prevent getting the data segment value in the high word of the value when casting a handle or near pointer value to `LRESULT` or `LPARAM`.
- Cast a far pointer, `LPARAM` or `LRESULT`, to a `DWORD` type and then to the desired type when you cast the far pointer to a handle or near pointer. This prevents "segment lost in conversion" warnings.
- Make sure you have the following lines, in the given order, in each source file:

```
#define STRICT
#include <windows.h>
```

- Compile your source to use the highest level of error checking. Treat any warnings as errors and correct your sources to eliminate the warning messages.
- Link and run the application to ensure that it executes without errors.

### Testing and Debugging Your Application in Windows

One of the advantages of an operating system such as Windows is its ability to run more than one application at a time. However, this advantage can also create hazards when you are testing and debugging an application.

Windows is a robust operating system. When Windows is running in protected (standard or 386 enhanced) mode, it can usually terminate an application that encounters a fatal error (such as an invalid handle) without affecting other applications. A fatal error or even a GP fault in an application very rarely causes the entire system to crash. However, it is possible to cause system failure in other ways when you are testing and debugging an application.

Because of the risk of system failure, you should always save all file buffers to disk before testing and



debugging your application. You should also avoid running other applications while testing and debugging your application if a general system failure would cause problems for the other applications.

#### Using Different Windows Versions

The Windows 3.1 SDK provides two environments for debugging or testing your Windows applications: a debugging version of the retail Windows product and a nondebugging version of the retail Windows product.

The SDK installation program creates two directories to contain the debugging and nondebugging versions of the core DLLs. Unless you specify different paths, Install places the debugging versions of the Windows core libraries in the \WINDEV\DEBUG directory and the nondebugging versions in the \WINDEV\NODEBUG directory. Install copies the nondebugging version of the Windows core libraries from your Windows system directory to the \WINDEV\NODEBUG directory.

You can conveniently switch between the debugging and nondebugging versions of Windows by running one of two batch files that Install places in the Windows development directory (named \WINDEV by default). The N2D.BAT file switches from the nondebugging to the debugging version, and D2N.BAT switches from the debugging to the nondebugging version.

These batch files either copy files from your \WINDEV\DEBUG and \WINDEV\NODEBUG directories or rename files in your Windows system directory. When you install the SDK files, Install asks if you want to keep a duplicate set of the libraries and symbol files in your Windows system directory. If you answer Yes, N2D.BAT and D2N.BAT quickly rename the duplicate files. Otherwise, the batch files copy the DLLs to your Windows system directory from the appropriate directory.

If you choose to retain a duplicate set of files, the DLLs and symbol files for the two versions of Windows appear in your Windows system directory with the same names as the core libraries and symbol files, but with the letter N (nondebugging) or D (debugging) appended to the name. For example, in addition to the GDI.EXE file, your system directory will contain the GDID.EXE and GDIN.EXE files.

#### Debugging Version

The debugging version of Windows consists of a set of DLLs that replace the Windows core DLLs of the retail product. The replaced DLLs are KRNL286.EXE, KRNL386.EXE, USER.EXE, GDI.EXE, and MMSYSTEM.DLL. Accompanying these DLLs is a set of symbol (.SYM) files.

The debugging versions of the core DLLs provide error checking and diagnostic messages that help you debug a Windows application. The symbol-file information helps you track calls into Windows when using the Microsoft Windows 80386 Debugger (WDEB386.EXE). In addition, the debugging versions of these DLLs contain Microsoft®CodeView®symbol information for tracking calls into Windows when using Microsoft®CodeView®for Windows™ (CVW).

A special setting is available in the [386Enh] section of SYSTEM.INI for the debugging version of Windows. The form of this setting follows:

```
DebugPhysAddr = {TRUE|FALSE}
```

By default, Windows makes the entire base physical linear memory region available when a debugger is loaded. Setting the DebugPhysAddr option to FALSE overrides this default when the debugger is loaded. Although the FALSE setting prevents you from being able to examine all memory, it creates a memory environment more like the nondebugging version of Windows, which can help you spot problems with pointers more quickly. The default value for DebugPhysAddr is TRUE.

#### Nondebugging Version

During application development, you should use the debugging version of Windows. However, use the nondebugging version of Windows whenever you want to do the following:

- Test the final version of your application
- Test the performance of your application without the performance disadvantages of the debugging version of Windows

Use the nondebugging version of Windows with the core DLLs supplied by the retail version of Windows. The Windows 3.1 SDK also provides symbol files for the nondebugging version of Windows. The retail Windows core libraries do not contain CodeView symbol information, however.

#### Using the System Debugging Log Application

The System Debugging Log Application (DBWIN.EXE) allows you to display messages produced by the debugging version of Windows even if you are not running a debugger and do not have a debugging

terminal. DBWIN.EXE allows you to control the output of specific types of messages. It also includes a feature that forces memory-allocation errors when testing the robustness of an application.

Note: DBWIN.EXE can provide useful debugging messages with the retail version of Windows as well. When you run DBWIN.EXE with the retail version of Windows, the Settings and Alloc Break commands are disabled in the Options menu, and you will only see a limited subset of debugging messages.

### System Debugging Output

The default system debugging output goes to AUX. DBWIN.EXE can also send debugging messages to COM1 or COM2. Sending debugging output to COM1 or COM2 improves the performance of your debugging system when you have redirected system debugging output to NUL, or if DBWIN.EXE is not running.

To disable AUX as the default, add the following setting to the [Debug] section of SYSTEM.INI:

```
OutputTo=NUL
```

To disable the default kernel output and to send output to COM1 or COM2, set the MS-DOS COM port baud rates to match the baud rates of your debugging terminal by using the MS-DOS mode command. To ensure that the settings are always correct, use the mode command in your AUTOEXEC.BAT file.

You can log messages to the system debugging window, to a monochrome screen, or to the COM1 or COM2 devices. The default destination for messages is to a window.

You can choose different destinations for debugging messages from the Options menu. These settings stay in effect the next time you run DBWIN.EXE.

### System Debug Options Dialog Box

When you choose the Settings command on the Options menu, a System Debug Options dialog box appears. This dialog box allows you to control the output of debugging messages produced by the debugging version of Windows.

The System Debug Options dialog box works only when you are running the debugging version of Windows. There are three groups of check boxes, described as follows:

- Break Options           Control whether and how a message will cause a break and stack trace to the debugger.
- Debug Options           Control the kind of debugging features that are enabled in the system.
- Trace Options           Control whether or not certain kinds of informational messages are produced.

The check boxes for Break Options and Trace Options are self-explanatory. The following list explains the check boxes for the Debug Options:

Option	Description
Validate Heap	Checks the consistency of global and local heaps before every call to a memory-management function. This option affects the global heap only when it is one of the default startup settings (that is, when it is saved by choosing the Save Settings command on the File menu). This option affects local heaps only if it is set before the application is started.
Check Free Blocks	Ensures that freed local blocks are not written into. The value 0xFB is written into free blocks, and when the heap is validated, a check is performed to ensure that the blocks are still filled with this value. This option works only with local heaps. This option must be used with the Validate Heap option.
Buffer Fill	Fills buffers that are passed to Windows functions with the value 0xF9. This option ensures that all of the supplied buffer is writable and helps detect overwrite problems that can occur when the buffer is too small.
Break with INT 3	Breaks to the debugger with an int 3 instruction, instead of a fatal exit. This option does not display a stack trace.
Don't trap faults	Prevents the system from hooking GP and stack overflow faults. (Many faults that result from choosing this option would normally be handled by the system. Choosing this option results in faults that would not occur otherwise.)

### Alloc Break Command

The Alloc Break command on the Options menu ensures that an application deals properly with out-of-memory conditions. This command displays a dialog box into which you can enter the module name of

your application and the number of memory allocations you want to succeed before subsequent allocations fail.

The system counts each global or local memory allocation performed by your application. When the number of allocations reaches the allocation break count, that allocation and all subsequent allocations fail. Because memory allocations made by the system fail once the break count is reached, calls to certain functions (such as `CreateWindow` and `SelectObject`) fail as well. Only allocations made within the context of the application you specify are affected by the allocation break count.

The module name is limited to 8 characters. In some cases the module name may be different from the filename; the module name is specified in the module-definition (.DEF) file for the application. You cannot specify the module name of a DLL.

If you set the break count to zero, no allocation break is set, but the system counts allocations made by the specified application. You can choose the Show Count button to display the current allocation count.

You can set an allocation break before an application is run. The allocation count is then set to zero and allocations are counted as soon as the application starts. If you run more than one instance of an application, the allocation break applies only to the most recent instance.

The allocation count is also reset to zero when you choose the Set or the Inc & Set button. You can set an allocation break before performing an operation to ensure that your application handles the problem effectively. Then you can choose Inc & Set and repeat the operation to ensure that the next allocation failure is also handled properly.

## Control Panel Applications

This topic describes Control Panel (CONTROL.EXE) for the Microsoft Windows operating system. It explains how to create a Control Panel application and then add the application to Control Panel.

Control Panel provides a window for running applications. These applications are used to configure the Windows environment. A number of standard applications are included with Windows. However, additional ones can be created and added to Control Panel. This capability is useful for modifying environmental factors unique to specific hardware and software.

An application is contained in a dynamic-link library (DLL). A DLL can support more than one Control Panel application.

Control Panel loads Control Panel application libraries in this order:

- 1 The library containing the standard Control Panel applications
- 2 Libraries specified in the [MMCPL] section of the CONTROL.INI file
- 3 Libraries with the .CPL filename extension residing in the same directory as the CONTROL.EXE file
- 4 Libraries with the .CPL filename extension residing in the Windows SYSTEM directory

### Starting a Control Panel Application

There are three ways to start a Control Panel application:

- The user can open Control Panel and start an application by double-clicking the application icon.
- The user or an application can open Control Panel by using a command-line argument that specifies the name of the application to start. When the Control Panel application closes, Control Panel automatically closes.
- An application can send a WM\_CPL\_LAUNCH message to Control Panel while Control Panel is running. When the Control Panel application closes, Control Panel sends back a WM\_CPL\_LAUNCHED confirmation message.

The following example shows how an application can start Control Panel and the Printers application from the command line by using the WinExec function:

```
WinExec("control.exe printers", SW_SHOWNORMAL)
```

When Control Panel starts, it immediately displays the Printers application. After the Printers application finishes, Control Panel ends.

The following example shows a function that starts a Control Panel application by using the WM\_CPL\_LAUNCH message:

```
BOOL StartApplet(LPSTR lpszName, HWND hwndMine)
{
    HGLOBAL hglbAppletName; /* global-object handle for app name */
    /
    HWND hwndCPL; /* handle of Control Panel window */
    LPSTR lpszAppletName; /* name of the application */
    BOOL fStartedCPL = FALSE; /* application started by CONTROL.EXE? */
    /

    /*
    * Allocate a global, shareable memory block to hold the
    * application-name string.
    */

    hglbAppletName = GlobalAlloc(GMEM_MOVEABLE, lstrlen(lpszName) + 1)
    ;
    if(hglbAppletName == NULL)
        return FALSE;
    lpszAppletName = GlobalLock(hglbAppletName);
    lstrcpy(lpszAppletName, lpszName);
    GlobalUnlock(hglbAppletName);

    /*
    * Get the Control Panel window handle and start Control Panel, if
    * necessary.
    */
}
```

```

*/

    if((hwndCPL = FindWindow("CtlPanelClass",
"Control Panel")) == NULL) {
WinExec("control.exe", SW_SHOWNA);
hwndCPL = FindWindow("CtlPanelClass", "Control Panel");
if(hwndCPL == NULL) {
GlobalFree(hglbAppletName);
return FALSE;
}
fStartedCPL = TRUE;
}

/* Start the application and end Control Panel, if started. */

SendMessage(hwndCPL, WM_CPL_LAUNCH, (WPARAM) hwndMine,
(LPARAM) lpszAppletName);
if (fStartedCPL)
SendMessage(hwndCPL, WM_CLOSE, 0, 0);
GlobalFree(hglbAppletName);
return TRUE;
}

```

### Creating a Control Panel Application

A Control Panel application must reside in a DLL that includes a standard entry-point function named **CPIApplet**. The application must include the **CPL.H** header file for the definition of the Control Panel messages. Control Panel communicates with the DLL by sending the following CPL messages to the **CPIApplet** function:

Message	Description
<b>CPL_DBLCLK</b>	Sent when the user double-clicks an application icon. In response to this message, the DLL should start its configuration process, usually displaying a dialog box.
<b>CPL_EXIT</b>	Sent after the last <b>CPL_STOP</b> message and immediately before Control Panel calls the <b>FreeLibrary</b> function for the DLL. In response to this message, the DLL should free any remaining memory and prepare to exit.
<b>CPL_GETCOUNT</b>	Sent after the <b>CPL_INIT</b> message, to prompt the DLL to return a number indicating how many applications it services.
<b>CPL_INIT</b>	Sent immediately after the DLL is loaded, to prompt the DLL to perform initialization procedures, including memory allocation.
<b>CPL_INQUIRE</b>	Sent after the <b>CPL_GETCOUNT</b> message, to prompt the DLL to provide information about each application. The handler for this message is a good place to include any initialization required by individual applications.
<b>CPL_NEWINQUIRE</b>	Sent to a Control Panel DLL to request information about an application that the DLL supports. The <b>CPL_NEWINQUIRE</b> message is the same as the <b>CPL_INQUIRE</b> message except that its second parameter ( <b>IPParam2</b> ) is a pointer to a <b>NEWCPLINFO</b> structure instead of a <b>CPLINFO</b> structure. New applications should use <b>CPL_NEWINQUIRE</b> instead of <b>CPL_INQUIRE</b> .
<b>CPL_SELECT</b>	Sent when the user selects an application icon.
<b>CPL_STOP</b>	Sent once for each application before Control Panel ends. In response to this message, the DLL should free any memory associated with the individual application for which the message is sent.

### Creating the Entry-Point Function

Control Panel communicates with an application DLL through the **CPIApplet** function. Be sure to export this function by listing it in the **EXPORTS** statement of your module-definition (**.DEF**) file. The **CPIApplet** function handles the messages listed previously, performing three main tasks:

#### Task Result

##### Initializing the application (**CPL\_INIT**, **CPL\_INQUIRE**)

Allocates any memory needed and gives Control Panel the information needed to display the application icon.

##### Running the application (**CPL\_DBLCLK**)

Passes control to a dialog box and its associated message processor.  
Closing the application (CPL\_STOP, CPL\_EXIT)  
Frees any memory allocated and prepares to exit.

The CPIApplet function has the following format:

LONG CALLBACK\* CPIApplet(hwndCPL, msg, lParam1, lParam2)

The hwndCPL parameter contains the handle of the Control Panel window. The msg parameter contains one of the CPL messages listed previously. The lParam1 and lParam2 parameters contain message-dependent values.

Initializing the Application

To initialize a Control Panel application, Control Panel sends the CPL\_INIT message first to the CPIApplet function, which prompts the application DLL to perform initialization procedures. If initialization succeeds, CPIApplet returns nonzero.

If CPIApplet returns zero in response to the CPL\_INIT message, Control Panel calls the FreeLibrary function and ends communication with the application DLL. This is the only way an application can notify Control Panel of initialization problems and prevent the application from being loaded.

If initialization is successful, Control Panel sends the CPL\_GETCOUNT message. The CPIApplet function responds by returning the number of applications serviced by the application DLL. This number determines how many icons Control Panel displays for the DLL.

Once Control Panel finds out the number of applications serviced by the DLL, it sends the CPL\_NEWINQUIRE message once for each application. The lParam1 parameter specifies the application number, which is zero for the first application and CPL\_GETCOUNT minus 1 for the last application.

Control Panel passes a far pointer to a NEWCPLINFO structure in the lParam2 parameter. The NEWCPLINFO structure has the following form:

```
typedef struct tagNEWCPLINFO /* ncpli */
{
    DWORD dwSize; /* length of structure, in bytes */
    DWORD dwFlags; /* setup flags */
    DWORD dwHelpContext; /* help-context number */
    LONG lData; /* application-defined data */
    HICON hIcon; /* handle of icon (owned by CPL.EXE) */
    char szName[32]; /* short-name string */
    char szInfo[64]; /* description string (status line) */
    char szHelpFile[128]; /* path to help file */
} NEWCPLINFO;
```

The CPIApplet function must fill in the NEWCPLINFO structure. The function must assign values for the dwSize, hIcon, szName, and szInfo members for the structure size, application icon, short name, and description. To add an accelerator key for the application, precede the selected accelerator character in the szName string with an ampersand. If the application DLL supports context-sensitive Help, the CPIApplet function should also assign the values for the dwHelpContext and szHelpFile members. The lData member can be used for application-defined data.

Note: The CPL\_NEWINQUIRE message and NEWCPLINFO structure replace the CPL\_INQUIRE message and CPLINFO structure. The latter have been kept for backward compatibility with Windows version 3.0. If the application DLL does not respond to the CPL\_NEWINQUIRE message, Control Panel sends it the CPL\_INQUIRE message. Then the lParam2 parameter points to a CPLINFO structure rather than to a NEWCPLINFO structure.

Responding to User Actions

Control Panel sends the CPL\_SELECT and CPL\_DBLCLK messages when the user selects (single-clicks) or double-clicks an application icon. For each message, Control Panel passes the application number in lParam1 and the lData value in lParam2.

Typically, an application DLL responds to the CPL\_SELECT message by doing nothing. When it receives the CPL\_DBLCLK message, it transfers control to the appropriate dialog box.

Exiting the Application and the DLL

Before exiting, Control Panel sends the CPL\_STOP message once for each application in the DLL. The lParam1 and lParam2 parameters sent with the CPL\_STOP message correspond to the application number and the IData value. After Control Panel sends the last CPL\_STOP message, it sends a CPL\_EXIT message and then calls the FreeLibrary function to free the DLL.

When the CPL\_STOP and CPL\_EXIT cases in the switch statement are executed, the DLL frees memory that it allocated. Typically, the DLL frees memory associated with individual applications when the CPL\_STOP case is executed and frees any other allocated memory when the CPL\_EXIT case is executed.

#### Example of a Control Panel Application

The following example shows the CPlApplet function for a DLL containing three Control Panel applications that set preferences for a component stereo system attached to the computer.

The example uses a programmer-defined StereoApplets array that contains three structures, each corresponding to one of the Control Panel applications. Each structure contains all the information required by the CPL\_INQUIRE message, as well as the dialog box template and dialog box procedure required by the CPL\_DBLCLK message. The following example fills the structures in the StereoApplets array:

```
#define NUM_APPLETS 3

typedef struct tagApplets
{
    int icon; /* icon-resource identifier */
    int namestring; /* name-string resource identifier */
    int descstring; /* description-string resource identifier */
    int dlgtemplate; /* dialog box template resource identifier */
    DLGPROC dlgfn; /* dialog box procedure*/
} APPLETS;

APPLETS StereoApplets[NUM_APPLETS] =
{
    AMP_ICON, AMP_NAME, AMP_DESC, AMP_DLG, AmpDlgProc,
    TUNER_ICON, TUNER_NAME, TUNER_DESC, TUNER_DLG, TunerDlgProc,
    TAPE_ICON, TAPE_NAME, TAPE_DESC, TAPE_DLG, TapeDlgProc,
};
```

This code defines the CPlApplet function for the preceding example:

```
LONG CALLBACK CPlApplet(hwndCPL, msg, lParam1, lParam2)
HWND hwndCPL; /* handle of Control Panel window */
UINT msg; /* message */
LPARAM lParam1; /* first message parameter */
LPARAM lParam2; /* second message parameter */
{
    int i;
    LPCPLINFO lpCplInfo;

    i = (int) lParam1;

    switch (msg) {
        case CPL_INIT: /* first message, sent once */
            return TRUE;

        case CPL_GETCOUNT: /* second message, sent once */
            return NUM_APPLETS;
            break;

        case CPL_INQUIRE: /* third message, sent once per app */
            lpCplInfo = (LPCPLINFO) lParam2;

            lpCplInfo->idIcon = StereoApplets[i].icon;
            lpCplInfo->idName = StereoApplets[i].namestring;
            lpCplInfo->idInfo = StereoApplets[i].descstring;
            lpCplInfo->lData = 0;
    }
```

```

break;

    case CPL_SELECT: /* application selected */
break;

    case CPL_DBLCLK: /* application double-clicked */
    DialogBox(hinst,
    MAKEINTRESOURCE(StereoApplets[i].dlgtemplate),
    hwndCPL, StereoApplets[i].dlgfn);
break;

    case CPL_STOP: /* sent once per app before CPL_EXIT */
break;

    case CPL_EXIT: /* sent once before FreeLibrary called */
break;

    default:
break;
    }
    return 0;
}

```

### Installing a New Application

There are three ways to register an application DLL with Control Panel:

- List the DLL in the [MMCPL] section of the CONTROL.INI file. Use this method when the DLL is part of a system library and handles more than just messages from Control Panel. The following is a sample CONTROL.INI entry:

```

[MMCPL]
myapplets=mydll.dll

```

- Assign the DLL a .CPL filename extension and install it in the directory that contains the CONTROL.INI file.
- Assign the DLL a .CPL filename extension and install it in the Windows SYSTEM directory.

See Also  
CPLApplet



## File Manager Extensions (3.1)

This topic describes how to create and install extensions for File Manager in the Microsoft Windows operating system. A File Manager extension is a dynamic-link library (DLL) that adds a menu to File Manager.

File Manager maintains a list of extensions in an initialization file and loads the extensions when starting. An extension DLL contains an entry point that processes menu commands and notification messages sent by File Manager. Up to five extension DLLs can be installed at any one time.

### Creating a File Manager Extension

A File Manager extension must reside in a DLL that includes a standard entry point, the `FMExtensionProc` function. It must include the `WFEXT.H` header file that defines File Manager messages and structures. File Manager communicates with the extension DLL by sending the following messages to the DLL's `FMExtensionProc` function:

Message	Meaning
1 through 99	User has selected an item from the extension-supplied menu. The value is the identifier of the selected menu item.
<code>FMEVENT_INITMENU</code>	User has selected the extension's menu. The extension should initialize items in the menu.
<code>FMEVENT_LOAD</code>	File Manager is loading the extension DLL and prompts the DLL for information about the menu that the DLL supplies.
<code>FMEVENT_SELCHANGE</code>	Selection in the File Manager directory window or Search Results window has changed.
<code>FMEVENT_UNLOAD</code>	Extension DLL is being unloaded.
<code>FMEVENT_USER_REFRESH</code>	User has chosen the Refresh command from the Window menu. The extension should update items in the menu, if necessary.

### Creating the Entry-Point Function

File Manager communicates with an extension DLL through the `FMExtensionProc` function. Be sure to export this function by listing it in an `EXPORTS` statement of your module-definition (`.DEF`) file. The `FMExtensionProc` function handles the messages listed in the previous section, performing the following tasks:

#### Task Action

##### Initializing the extension (`FMEVENT_LOAD`)

Provides File Manager with the name and handle of the menu and saves the menu-item delta value.

##### Initializing the menu (`FMEVENT_INITMENU`)

Initializes all top-level menu items and the items in any submenus.

##### Processing menu selections

Carries out commands that the user chooses from the extension's menu.

##### Processing file selections (`FMEVENT_SELCHANGE`)

Queries File Manager for information about the file that the user has selected from the directory window or Search Results window.

##### Updating items in the menu (`FMEVENT_USER_REFRESH`)

Modifies the menu as appropriate when the user chooses File Manager's Refresh command from the Window menu.

##### Quitting the extension DLL (`FMEVENT_UNLOAD`)

Frees any memory allocated and prepares to exit.

The `FMExtensionProc` function is defined as follows:

```
HMENU CALLBACK FMExtensionProc(hwnd, msg, lParam)
HWND hwnd;
UINT msg;
LPARAM lParam;
```

The `hwnd` parameter identifies the File Manager window. An extension should use this window handle to specify the parent window for any dialog boxes or message boxes it needs to display. It should also use this handle to send query messages to File Manager. The `msg` parameter contains one of the File Manager messages listed previously. The `lParam` parameter contains a message-dependent value. The return value from the `FMExtensionProc` function depends on the value of the `msg` parameter.

The menu added to File Manager may be a hierarchical (cascaded) menu and may contain grayed, disabled, or checked menu items in addition to command items. Menu items should be text only; owner-drawn menus and bitmap menus are not supported. Changing the check-mark bitmap is not supported.

Whenever File Manager calls the `FMExtensionProc` function, it waits to refresh its directory windows (for changes in the file system) until after the function returns. This allows the extension to perform large numbers of file operations without excessive repainting on the part of File Manager. The extension does not need to send the `FM_REFRESH_WINDOWS` message to notify File Manager to repaint its directory windows.

#### Loading the Extension

File Manager sends, first, the `FMEVENT_LOAD` message to the `FMExtensionProc` function. The `IParam` parameter that accompanies the `FMEVENT_LOAD` message points to an `FMS_LOAD` structure that File Manager uses to obtain information about the extension-supplied menu, including the menu name and menu handle.

File Manager also uses the `FMS_LOAD` structure to pass the menu-item delta value to the extension. To avoid conflicts with its own menu-item identifiers, File Manager rennumbers the menu-item identifiers in an extension-supplied menu by adding the delta value to each identifier. If an extension DLL needs to modify its menu after File Manager has loaded it, it must use the delta value. For example, to delete a menu item, the extension DLL finds the sum of the delta value and the menu item's identifier and then passes the sum as the `idItem` parameter to the `DeleteMenu` function.

#### Processing Menu Selections

The menu resource that you define for your extension's menu must use menu-item identifiers in the range 1 through 99. When the user selects an item, the extension receives a command notification, which is the actual identifier of the selected item as defined in the resource-definition file (which has the `.RC` filename extension). The command notification is not the sum of the delta value and the identifier. An extension DLL's `FMExtensionProc` function carries out commands by processing command notifications.

#### Initializing the Extension Menu

Whenever the user selects the extension's main menu item from File Manager's menu bar, File Manager sends the `FMEVENT_INITMENU` message to the extension DLL. An extension can use this message to initialize its menu items. For example, an extension can add check marks, disable items, or gray items during this message.

When the user selects submenus within the extension's menu, File Manager does not send the `FMEVENT_INITMENU` message. An extension DLL must initialize all items at the same time, including those in submenus.

#### Updating the Extension Menu

When the user chooses the Refresh command from the Window menu, File Manager sends an `FMEVENT_USER_REFRESH` message to an extension DLL. An extension can use this opportunity to update its menu items.

#### Processing File Selections

When the user selects a filename in the directory window or in the Search Results window, File Manager sends the `FMEVENT_SELCHANGE` message to an extension DLL. An extension can use this opportunity to send a query message to File Manager to obtain more information about the user's selection. For more information, see Extension Messages.

Because the user can change the selection often, the extension should return promptly after processing the `FMEVENT_SELCHANGE` message to avoid slowing the user's selection process.

#### Quitting the Extension DLL

When File Manager quits, it sends the `FMEVENT_UNLOAD` message to each extension DLL and then calls the `FreeLibrary` function to free the DLLs. Each DLL should free any memory that it has allocated.

#### Installing Extensions

File Manager installs extensions that have settings in the [AddOns] section of the `WINFILE.INI` initialization file. Each setting contains an entry and a value. An entry consists of a string that represents the name of an extension. The value assigned to the entry consists of a string that specifies the path to the extension DLL. An application can use the `WritePrivateProfileString` function to add a setting to `WINFILE.INI`. The following example shows a setting in `WINFILE.INI`:

```
[AddOns]
My File Manager Extension=c:\win\system\rfmine.dll
```

File Manager does not display an error message if it cannot find an extension DLL, so an extension DLL can be deleted in order to uninstall it. Even so, an application that installs an extension DLL should provide an uninstall option to remove the extension's setting from the WINFILE.INI file.

#### Extension Messages

An extension can send the following window messages to retrieve relevant information from File Manager. File Manager is only guaranteed to respond correctly to messages sent from the FMExtensionProc function.

Message	Description
FM_GETDRIVEINFO	File Manager returns drive information from the active window. An extension provides a pointer to an <u>FMS_GETDRIVEINFO</u> structure; File Manager fills the structure with drive information.
FM_GETFILESEL	File Manager returns information about a selected file from the active File Manager window (either the directory window or the Search Results window). An extension provides a pointer to an <u>FMS_GETFILESEL</u> structure; File Manager fills the structure with file information.
FM_GETFILESELLFN	Same as the <u>FM_GETFILESEL</u> message except that the selected file may have a long filename.
FM_GETFOCUS	File Manager returns a value that identifies the type of window with input focus.
FM_GETSELCOUNT	File Manager returns the count of selected files in the directory and Search Results windows.
FM_GETSELCOUNTLFN	Same as the <u>FM_GETSELCOUNT</u> message except that the count includes files with long filenames.
FM_REFRESH_WINDOWS	File Manager repaints either its active window or all of its windows. This message is similar to File Manager's Refresh command on the Window menu.
FM_RELOAD_EXTENSIONS	File Manager reloads all extensions. First File Manager unloads all extensions, sending an <u>FMEVENT_UNLOAD</u> message to each extension. Then it reloads the extensions, sending an <u>FMEVENT_LOAD</u> message to each extension. The <u>FM_RELOAD_EXTENSIONS</u> message allows an extension to uninstall itself by removing its setting from the WINFILE.INI file; this action causes File Manager to reload the remaining extensions. Other applications (for example, installation programs) can also post this message by calling the <u>PostMessage</u> function.

#### File Manager Extension Example

The following example shows the FMExtensionProc function for a sample extension DLL. It demonstrates how an extension processes the menu commands and notification messages sent by File Manager.

```
HINSTANCE hinst;
HMENU hmenu;
WORD wMenuDelta;
BOOL fMultiple = FALSE;
BOOL fLFN = FALSE;

DWORD FAR PASCAL FMExtensionProc(HWND hwnd, WPARAM wParam, LPARAM lParam)
HWND hwnd;
WORD wParam;
LONG lParam;
{
    char szBuf[200];
    int count;
```

```

    switch (wMsg) {
    case FMEVENT_LOAD:

#define lpload ((LPFMS_LOAD)lParam)

/* Save the menu-item delta value. */

wMenuDelta = lpload->wMenuDelta;

/* Fill the FMS_LOAD structure. */

lpload->dwSize = sizeof(FMS_LOAD);
lstrcpy(lpload->szMenuName, "&Extension");

/* Return the handle of the menu. */

return (DWORD) (lpload->hMenu = LoadMenu(hinst,
MAKEINTRESOURCE(MYMENU)));
break;

    case FMEVENT_UNLOAD:

/* Perform any cleanup procedures here. */

break;

    case FMEVENT_INITMENU:

/* Copy the menu-item delta value and menu handle. */

wMenuDelta = LOWORD(lParam);
hmenu = (HMENU) HIWORD(lParam);

/*
 * Add check marks to menu items as appropriate. Add menu-
 * item delta values to menu-item identifiers to specify the
 * menu items to check.
 */

CheckMenuItem(hmenu, wMenuDelta + MULTIPLE,
fMultiple ? MF_BYCOMMAND | MF_CHECKED :
MF_BYCOMMAND | MF_UNCHECKED);
CheckMenuItem(hmenu, wMenuDelta + LFN,
fLFN ? MF_BYCOMMAND | MF_CHECKED :
MF_BYCOMMAND | MF_UNCHECKED);
break;

    case FMEVENT_USER_REFRESH:
MessageBox(hwnd, "User refresh event", "Hey!", MB_OK);
break;

    case FMEVENT_SELCHANGE:
OutputDebugString("Sel change\r\n");
break;

/*
 * The following messages are generated when the user chooses
 * items from the extension menu.
 */

    case GETFOCUS:
wsprintf(szBuf, "Focus %d", (int)SendMessage(hwnd,
FM_GETFOCUS, 0, 0L));
MessageBox(hwnd, szBuf, "Focus", MB_OK);

```

```

break;

    case GETCOUNT:
        count = (int)SendMessage(hwnd,
fLFN ? FM_GETSELCOUNTLFN : FM_GETSELCOUNT, 0, 0L);

        wsprintf(szBuf, "%d files selected", count);
MessageBox(hwnd, szBuf, "Selection Count", MB_OK);
        break;

    case GETFILE:
        {
FMS_GETFILESEL file;

        count = (int) SendMessage(hwnd,
fLFN ? FM_GETSELCOUNTLFN : FM_GETSELCOUNT,
FMFOCUS_DIR, 0L);

        while (count >= 1) {

/* Selection indices are zero-based (0 is first). */

count--;
SendMessage(hwnd, FM_GETFILESEL, count,
(LONG) (LPFMS_GETFILESEL)&file);
OemToAnsi(file.szName, file.szName);
wsprintf(szBuf, "file %s\nSize %ld",
(LPSTR)file.szName, file.dwSize);
MessageBox(hwnd, szBuf, "File Information", MB_OK);

if (!fMultiple)
    break;
        }
        break;

    case GETDRIVE:
        {
FMS_GETDRIVEINFO drive;

SendMessage(hwnd, FM_GETDRIVEINFO, 0,
(LONG) (LPFMS_GETDRIVEINFO)&drive);

OemToAnsi(drive.szVolume, drive.szVolume);
OemToAnsi(drive.szShare, drive.szShare);

wsprintf(szBuf,
"%s\nFree Space %ld\nTotal Space %ld\nVolume %s\nShare %s",
(LPSTR) drive.szPath, drive.dwFreeSpace,
drive.dwTotalSpace, (LPSTR) drive.szVolume,
(LPSTR) drive.szShare);
MessageBox(hwnd, szBuf, "Drive Info", MB_OK);
        break;
        }

    case LFN:
        fLFN = !fLFN;
        break;

    case MULTIPLE:
        fMultiple = !fMultiple;
        break;

    case REFRESH:

```

```

    case REFRESHALL:
        SendMessage(hwnd, FM_REFRESH_WINDOWS,
        wParam == REFRESHALL, 0L);
        break;

    case RELOAD:
        PostMessage(hwnd, FM_RELOAD_EXTENSIONS, 0, 0L);
        break;
    }
    return NULL;
}

```

#### Adding the Undelete Command

File Manager supports a hook for adding an Undelete command to the File menu (below the Delete command). If an undelete dynamic-link library is specified in the WINFILE.INI file, File Manager adds the Undelete command to the File menu when it starts. When the user chooses the Undelete command, File Manager calls the DLL.

The [settings] section of the WINFILE.INI file should include a reference to the undelete DLL, as follows:

```

[settings]
UNDELETE.DLL=C:\MYDIR\OTHER.DLL

```

An undelete DLL must include a standard entry point, the UndeleteFile function. This function must be exported by specifying the name of the function in the EXPORTS statement of the DLL's module-definition (.DEF) file.

The UndeleteFile function is defined as follows:

```

int CALLBACK UndeleteFile(hwndParent, lpszDir)
HWND hwndParent;
LPSTR lpszDir;

```

The hwndParent parameter identifies the parent window for any dialog boxes that the DLL creates. The lpszDir parameter specifies the initial directory to be used (for example, C:\TEMP).

See Also

File Manager Extension Functions, File Manager Extension Messages, File Manager Extension Structures

## Shell Dynamic-Data Exchange Interface Overview (3.1)

This topic describes the dynamic data exchange (DDE) interface of Windows Program Manager (PROGMAN.EXE). Program Manager is an application that lets users group, start, and otherwise control other applications for the Microsoft Windows operating system. Program Manager starts automatically when the user starts Windows and continues to run as long as Windows is in use. Upon starting, Program Manager displays one or more windows within its main window. Each window contains icons that correspond to logically related Windows applications. For example, the Main window contains an icon for the File Manager, Control Panel, Print Manager, Clipboard, MS-DOS Prompt, and Windows Setup applications.

The following topics are related to the information in this topic:

- Atoms
- Dynamic data exchange (DDE)
- Registration database

### PROGMAN.INI File

When Program Manager starts, it searches its initialization file for a list of group files. The windows that appear in Program Manager's main window correspond to group files. From the user's perspective, a group file is a collection of icons that represent logically related applications, but from the programmer's perspective, a group file is actually a collection of data. This data includes the color information for the icons (their AND and XOR masks), an offset to the resource header for each icon, the ideal resolution for displaying each icon, the name of the executable file that contains the application, and so on.

Group files are identified in the Program Manager initialization file (PROGMAN.INI). This initialization file has the following form:

```
[Settings]
Window=64 48 576 384 1
Order= 3 4 5 6 8 7 2 1 9
AutoArrange=1
SaveSettings=1
MinOnRun=1
Startup=
display.drv=v776816.drv

[Groups]
Group1=C:\WINDOWS\MAIN.GRP
Group2=C:\WINDOWS\ACCESSOR.GRP
Group3=C:\WINDOWS\GAMES.GRP
Group4=C:\WINDOWS\STARTUP.GRP
Group5=C:\WINDOWS\LZEXPAND.GRP
Group6=C:\WINDOWS\COMDLG.GRP
Group7=C:\WINDOWS\GDI.GRP
Group8=C:\WINDOWS\WINPROJ.GRP
Group9=C:\WINDOWS\MICROSOFT.GRP

[Restrictions]
NoRun=1
NoClose=1
NoSaveSettings=0
NoFileMenu=0
EditLevel=3
```

The following three sections describe the contents of the PROGMAN.INI file.

### Settings Section

The first section of the initialization file, [Settings], controls attributes of the Program Manager environment. The following entries appear in the [Settings] section:

Entry	Meaning
Window=	Specifies the location and dimensions of Program Manager's main window.
Order=	Specifies the order in which the groups listed in the [Groups] section appear in Program Manager's main window.

AutoArrange=	Specifies whether Program Manager should automatically arrange icons within groups.
SaveSettings=	Specifies whether to save the position of Program Manager's main window when exiting Program Manager.
MinOnRun=	Specifies whether to minimize Program Manager when an application is started.
Startup=	Specifies the name of the startup group. Program Manager automatically starts the applications in the startup group whenever it starts. If the startup group has a name other than "Startup", that name must be specified by the Startup= entry.
display.drv=	Specifies the display driver that was in use when Program Manager last ended. When Program Manager starts, it compares this value to the string in the SYSTEM.INI file. If they are different, Program Manager reextracts the application icons.

### Groups Section

The second section of the initialization file, [Groups], identifies the names of the group files for which Program Manager should display unique windows or icons. The groups must be numbered, but they need not be listed in any particular order. Program Manager never changes the number of an existing group, so if an application other than Program Manager constructs a PROGMAN.INI file, it can assign meaningful numbers to groups, if necessary.

### Restrictions Section

The third section of the initialization file, [Restrictions], disables some capabilities of the Program Manager environment. The following entries can appear in the [Restrictions] section:

Entry	Meaning												
NoRun=	Specifies whether to disable the Run command on the File menu. If this entry is set to 1, the command is disabled. If this entry is set to 0, the Run command is enabled. The default is 0 (enabled) if no value is specified.												
NoClose=	Specifies whether to prevent the user from exiting Program Manager through the File menu, the System menu, the ALT+F4 accelerator, or the Task List. If this entry is set to 1, exiting is prevented. If this entry is set to 0, exiting is allowed. The default is 0 (allowing exiting) if no value is specified.												
NoSaveSettings=	Specifies whether to disable the Save Settings on Exit command on the Options menu. If this entry is set to 1, the Save Settings on Exit command is disabled. If this entry is set to 0, the command is enabled. The default is 0 (enabled) if no value is specified.												
NoFileMenu=	Specifies whether to disable the File menu and all of its commands. If this entry is set to 1, the File menu is disabled. If this entry is set to 0, the menu is enabled. The default setting is 0 (enabled) if no value is specified.												
EditLevel=	Controls the extent to which the user can modify read-write groups. (Shared, read-only groups cannot be modified.) This entry may be set to one of the following values: <table> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Allows any modifications to the group. This is the default.</td> </tr> <tr> <td>1</td> <td>Prevents the user from creating, deleting, or renaming groups.</td> </tr> <tr> <td>2</td> <td>Prevents the user from creating, deleting, or renaming groups and from creating or deleting items in a group.</td> </tr> <tr> <td>3</td> <td>Prevents the user from creating, deleting, or renaming groups; from creating or deleting items in a group; and from changing command lines for items in a group.</td> </tr> <tr> <td>4</td> <td>Prevents the user from changing any property of an item in a group; from creating, deleting, or renaming groups; from creating or deleting items in a group; and from changing command lines for items in a group.</td> </tr> </tbody> </table>	Value	Meaning	0	Allows any modifications to the group. This is the default.	1	Prevents the user from creating, deleting, or renaming groups.	2	Prevents the user from creating, deleting, or renaming groups and from creating or deleting items in a group.	3	Prevents the user from creating, deleting, or renaming groups; from creating or deleting items in a group; and from changing command lines for items in a group.	4	Prevents the user from changing any property of an item in a group; from creating, deleting, or renaming groups; from creating or deleting items in a group; and from changing command lines for items in a group.
Value	Meaning												
0	Allows any modifications to the group. This is the default.												
1	Prevents the user from creating, deleting, or renaming groups.												
2	Prevents the user from creating, deleting, or renaming groups and from creating or deleting items in a group.												
3	Prevents the user from creating, deleting, or renaming groups; from creating or deleting items in a group; and from changing command lines for items in a group.												
4	Prevents the user from changing any property of an item in a group; from creating, deleting, or renaming groups; from creating or deleting items in a group; and from changing command lines for items in a group.												

Setting NoRun to 1 and EditLevel to 3 prevents a user from using Program Manager to run any applications that are not already in a program group.

### Command-String Interface

Program Manager has a DDE command-string interface that allows other applications to create, display, delete and reload groups; add items to groups; replace items in groups; delete items from groups; and to close Program Manager. The following commands perform these actions:



AddItem	ExitProgman
CreateGroup	Reload (Windows 3.1 only)
DeleteGroup	ReplaceItem (Windows 3.1 only)
<u>DeleteItem</u> (Windows version 3.1 only)	ShowGroup

The setup program for an application can use these commands, for example, to instruct Program Manager to install the application's icon in a group.

Multiple commands may be concatenated; each command must be contained in square brackets, and parameters must be contained in parentheses and separated by commas. Quotation marks must be used to delimit arguments that contain spaces, brackets, or parentheses. For example, the following set of commands adds WINAPP.EXE to the Windows Applications group:

```
[CreateGroup("Windows Applications")]
[ShowGroup("MYGROUP.GRP",1)]
[AddItem(winapp.exe,Win App,winapp.exe,2)]
```

To use these commands, an application must first initiate a conversation with Program Manager. The application and topic names for the conversation are both PROGMAN. Then the application sends the WM\_DDE\_EXECUTE message, specifying the appropriate command and its parameters.

Note: The user can configure Windows to use a shell other than Program Manager as the default. As a result, you should not design an application assuming that Program Manager will be available for a DDE conversation.

The following sections describe Program Manager DDE command strings in detail. In the syntax blocks in the following sections, brackets enclose optional arguments.

#### CreateGroup

The syntax for the CreateGroup command has this form:

```
CreateGroup(Group Name[,Group Path])
```

The CreateGroup command instructs Program Manager to create a new group or activate the window of an existing group.

Following are the parameters for this command:

Group Name	Identifies the group to be created. This parameter is a string. If a group already exists with the name specified by Group Name, CreateGroup activates the group window.
Group Path	Specifies the path of the group file. If your application does not supply this parameter, Windows uses a default filename for the group in the Windows directory.

#### ShowGroup

The syntax for the ShowGroup command has this form:

```
ShowGroup(Group Name,Show Command)
```

The ShowGroup command instructs Program Manager to minimize, maximize, or restore the window of an existing group.

Following are the parameters for this command:

Group Name	Identifies the group window to be minimized, maximized, or restored.
Show Command	Specifies the action that Program Manager is to perform on the group window. This parameter is an integer. It must have one of the following values:

Value	Meaning
1	Activates and displays the group window. If the window is minimized or maximized, Windows restores it to its original size and position.
2	Activates the group window and displays it as an icon.
3	Activates the group window and displays it as a maximized window.
4	Displays the group window in its most recent size and position. The window that is currently active remains active.
5	Activates the group window and displays it in its current size and position.
6	Minimizes the group window.
7	Displays the group window as an icon. The window that is currently active remains active.

Displays the group window in its current state. The window that is currently active remains active.

#### DeleteGroup

The syntax for the DeleteGroup command has this form:

```
DeleteGroup(GroupName)
```

The DeleteGroup command instructs Program Manager to delete an existing group.

Following is the parameter for this command:

GroupName Identifies the group to be deleted.

#### Reload

The syntax for the Reload command has this form:

```
Reload(GroupName)
```

The ReloadGroup command instructs Program Manager to remove and reload an existing group. An application that modifies group files can use this command to cause Program Manager to update the groups when it has finished making modifications.

Following is the parameter for this command:

GroupName Identifies the group to be removed and reloaded. If the GroupName parameter is not specified, Program Manager unloads all groups and reloads the [Group] section of PROGMAN.INI. The [Settings] and [Restrictions] sections are not reread.

#### AddItem

The syntax for the AddItem command has this form:

```
AddItem(CmdLine[,  
Name[,IconPath[,IconIndex[,xPos, yPos[,DefDir[,  
HotKey[,fMinimize] ] ] ] ] ] )
```

The AddItem command instructs Program Manager to add an icon to an existing group.

Following are the parameters for this command:

CmdLine	Specifies the full command line required to execute the application. This parameter is a string. At a minimum, this string is the name of the executable file for the application. It can also include the full path of the application and any parameters required by the application.
Name	Specifies the title that is displayed below the icon in the group window.
IconPath	Identifies the filename for the icon to be displayed in the group window. This parameter is a string. This file can be either a Windows executable file or an icon file. If the IconPath parameter is not specified, Program Manager uses the first icon in the file specified by the CmdLine parameter if that file is an executable file. If CmdLine specifies an associated file, Program Manager uses the first icon of the associated executable file. The association is taken from the registration database. (For more information about the registration database, see The Windows Shell Overview.) If CmdLine specifies neither an executable file nor an associated executable file, Program Manager uses a default icon.
IconIndex	Specifies the index of the icon in the file identified by the IconPath parameter. The IconIndex parameter is an integer. PROGMAN.EXE contains five built-in icons that can be used for non-Windows programs.
xPos	Specifies the horizontal position of the icon in the group window. This parameter is an integer. You must use both the xPos and yPos parameters to specify the position of the icon. If you do not specify the position, Program Manager places the icon in the next available space.
yPos	Specifies the vertical position of the icon in the group window. This parameter is an integer. You must use both the xPos and yPos parameters to specify the position of the icon. If you do not specify the position, Program Manager places the icon in the next available space.
DefDir	Specifies the name of the default (or working) directory. This parameter is a string.
HotKey	Identifies a hot (or shortcut) key that is specified by the user.
fMinimize	Specifies whether an application window should be minimized when it is first displayed.

## ReplaceItem

The syntax for the ReplaceItem command has this form:

ReplaceItem(ItemName)

The ReplaceItem command instructs Program Manager to delete an item and record the position of the deleted item. Program Manager will add a new item (specified by the next AddItem command) at this recorded position.

Following is the parameter for this command:

ItemName        Specifies the item to be deleted. Its position is recorded by Program Manager.

## DeleteItem

The syntax for the DeleteItem command has this form:

DeleteItem(ItemName)

The DeleteItem command instructs Program Manager to delete an item from the currently active group.

Following is the parameter for this command:

ItemName        Specifies the item to be deleted from the currently active group.

## ExitProgman

The syntax for the ExitProgman command has this form:

ExitProgman(bSaveGroups)

If Program Manager was started by another application, the ExitProgman command instructs Program Manager to exit and, optionally, save its group information.

Following is the parameter for this command:

bSaveGroups     Specifies a Boolean value that, if nonzero, causes Program Manager to save its group information before closing. If bSaveGroups is zero, Program Manager does not save its group information.

## Requesting Group Information

Program Manager can provide information about its groups to an application. Applications can request this information from Program Manager by using the PROGMAN topic.

An application can obtain a list of Program Manager groups by issuing a request for the Group item. Program Manager provides the list in CF\_TEXT format. The list consists of group-name strings separated by carriage returns.

An application can use a group name as an item name to request information about the group. Program Manager provides this information in CF\_TEXT format. The fields of group information are separated by commas. The first line of the information contains the group name (in quotation marks), the path of the group file, and the number of items in the group. Each subsequent line contains information about an item in the group, including the command line (in quotation marks), the default directory, the icon path, the position in the group, the icon index, the shortcut key (in numeric form), and the minimize flag.

## Dynamic Data Exchange Interface for Replacement Shells

You may choose to write an application that replaces the Windows shell. This replacement shell must be able to provide property information to the application that starts non-Windows programs in an MS-DOS window. (This application is known as WinOldApp.) This section discusses how a replacement shell can provide property information for WinOldApp. Applications other than WinOldApp do not need this information. The DDE protocol described in this section may not be supported in future versions of Windows.

## Properties

A replacement shell should maintain several pieces of information, called properties, for each application that WinOldApp might start. These are the same properties that appear in the Program Item Properties dialog box of Program Manager. These properties include:

- Description (title)
- Command line
- Working directory
- Shortcut key
- Icon

The shell must provide a DDE interface that allows WinOldApp to obtain three of these properties: description, working directory, and icon.

To obtain its properties from the shell, WinOldApp must accomplish the following tasks:

- 1 Establish a DDE conversation with the shell.
- 2 Request a property from the shell.
- 3 Receive a property from the shell.
- 4 Terminate the DDE conversation.

#### Establishing a DDE Conversation

WinOldApp requests property data from the shell by using the `SendMessage` function to broadcast the `WM_DDE_INITIATE` message. The `wParam` parameter of the `SendMessage` function is the handle of WinOldApp's DDE window. The low-order word of the `lParam` parameter is an atom that represents the name of the shell application: "Shell". The high-order word is an atom that represents the name of the properties topic: "AppProperties".

A "Shell" DDE server that supports the "AppProperties" topic responds to the `WM_DDE_INITIATE` message by sending a `WM_DDE_ACK` message. The server should send the following parameters with the `WM_DDE_ACK` message:

Parameter	Description
<code>hwnd</code>	Specifies the handle of WinOldApp's DDE window. The shell should use the handle that WinOldApp specified as the <code>wParam</code> parameter in the <code>WM_DDE_INITIATE</code> message.
<code>message</code>	Specifies the <code>WM_DDE_ACK</code> message.
<code>wParam</code>	Specifies the handle of the "Shell" server's DDE window.
<code>HIWORD(lParam)</code>	Specifies an atom that represents the name of the shell application: "Shell".
<code>LOWORD(lParam)</code>	Specifies an atom that represents the name of the properties topic: "AppProperties".

It is not necessary to free the atoms used in a conversation with WinOldApp. It is WinOldApp's responsibility to create and free the atoms.

#### Providing Property Data

After the DDE server that provides a replacement shell responds with a `WM_DDE_ACK` message to the `WM_DDE_INITIATE` from WinOldApp, WinOldApp sends a `WM_DDE_REQUEST` message to request property data. The server can respond to the `WM_DDE_REQUEST` message by posting a `WM_DDE_DATA` message.

The Windows shell associates an item name with each of the application properties that it provides. The item names are described in the following table:

Item name	Description
"GetDescription"	The shell provides an application's description (title) property.
"GetWorkingDIR"	The shell provides an application's working-directory property.
"GetIcon"	The shell provides an application's icon property.

WinOldApp requests properties by obtaining an atom for each of the item-name strings and passing the atoms to the shell in a sequence of `WM_DDE_REQUEST` messages (one message for each property). WinOldApp also passes the handle of the application's instance as the low-order word of the `lParam` parameter in the `WM_DDE_REQUEST` message. The shell should use this instance handle to find the properties associated with the application.

If a "Shell" DDE server does not recognize the application's instance handle, the server does not support properties for the application instance. In this case, the server should respond by sending a negative `WM_DDE_ACK` message. The parameters passed with the negative `WM_DDE_ACK` message are as follows:

Parameter	Description
<code>hwnd</code>	Specifies the handle of WinOldApp's DDE window. The shell should use the handle that WinOldApp specified as the <code>wParam</code> parameter in the <code>WM_DDE_REQUEST</code> message.
<code>message</code>	Specifies the <code>WM_DDE_ACK</code> message.
<code>wParam</code>	Specifies the handle of the "Shell" server's DDE window.

- LOWORD(IParam) Specifies zero. The "Shell" DDE server does not support properties for the specified application instance.
- HIWORD(IParam) Specifies an atom that represents the item name of the requested property. Depending on the type of property requested, the atom should identify one of the following strings: "GetDescription", "GetWorkingDIR", or "GetIcon".

When WinOldApp receives a negative WM\_DDE\_ACK message, it terminates the conversation with the "Shell" DDE server.

If a "Shell" DDE server recognizes the application's instance handle and the requested property is available, it should allocate a global memory object and copy the property data to the object. Then it should post a WM\_DDE\_DATA message to WinOldApp. The WM\_DDE\_DATA message should contain the handle of the global memory object.

The contents of the global memory object allocated by the shell depend on the type of property WinOldApp requested. The following three sections describe the description, working-directory, and icon properties.

#### Providing the Description Property

If the shell is responding to a request for the "GetDescription" property, the shell should pass the following parameters with the WM\_DDE\_DATA message:

Parameter	Description
hwnd	Specifies the handle of WinOldApp's DDE window. The shell should use the handle that WinOldApp specified as the wParam parameter in the <u>WM_DDE_REQUEST</u> message.
message	Specifies the <u>WM_DDE_DATA</u> message.
wParam	Specifies the handle of the shell's DDE window.
LOWORD(IParam)	Specifies a handle to a global memory object that contains a <u>DDEDATA</u> structure. A description of the contents of the <u>DDEDATA</u> structure follows this table. To report an error, the server should use one of the error values listed with the <u>WinExec</u> function.
HIWORD(IParam)	Specifies an atom that represents the string, "GetDescription".

The low-order word of the IParam parameter should be a handle to a global memory object that contains a DDEDATA structure (defined in the DDE.H header file). The contents of the DDEDATA structure are as follows:

```
#include <dde.h>

typedef struct tagDDEDATA { /* ddedat */
    WORD    unused:12,
    fResponse:1,
    fRelease:1,
    reserved:1,
    fAckReq:1;
    short   cfFormat;
    BYTE    Value[1];
} DDEDATA;
```

The Value member should contain the description property, in the form of a null-terminated string of characters from the Windows character set. The string can be any size but typically contains fewer than 30 characters.

If the server sets the fAckReq bit, WinOldApp responds to the WM\_DDE\_DATA message by posting a WM\_DDE\_ACK message after processing the data.

If the server sets the fRelease bit, WinOldApp frees the global memory object after copying the description string. Otherwise, WinOldApp does not free the memory object.

#### Providing the Working-Directory Property

If the shell is responding to WinOldApp's request for the "GetWorkingDIR" property, the shell passes the following parameters with the WM\_DDE\_DATA message:

Parameter	Description
hwnd	Specifies the handle of WinOldApp's DDE window. The shell should use the

handle that `WinOldApp` specified as the `wParam` parameter in the `WM_DDE_REQUEST` message.

message Specifies the `WM_DDE_DATA` message.

wParam Specifies the handle of the shell's DDE window.

LOWORD(IParam) Specifies a handle to a global memory object that contains a `DDEDATA` structure. A description of the contents of the `DDEDATA` structure follows this table. To report an error, the server should use one of the error values listed with the `WinExec` function.

HIWORD(IParam) Specifies an atom that represents the string, "GetWorkingDIR".

The low-order word of the `IParam` parameter is a handle to a global memory object that contains a `DDEDATA` structure. The contents of the `DDEDATA` structure are as follows:

```
#include <dde.h>

typedef struct tagDDEDATA { /* ddedat */
    WORD unused:12,
    fResponse:1,
    fRelease:1,
    reserved:1,
    fAckReq:1;
    short cfFormat;
    BYTE Value[1];
} DDEDATA;
```

The `Value` member should specify the location (drive and path) of the application's executable file, in the form of a null-terminated string of characters from the Windows character set. The character string has a maximum size of 128 characters (including the terminating null character).

If the server sets the `fAckReq` bit, `WinOldApp` responds to the `WM_DDE_DATA` message by posting a `WM_DDE_ACK` message after processing the working-directory property.

If the server sets the `fRelease` bit, `WinOldApp` frees the global memory object after copying the working-directory string. Otherwise, `WinOldApp` does not free the memory object.

#### Providing the Icon Property

If the shell is responding to `WinOldApp`'s request for "GetIcon" property, the shell passes the following parameters with the `WM_DDE_DATA` message:

Parameter	Description
hwnd	Specifies the handle of <code>WinOldApp</code> 's DDE window. The shell should use the handle that <code>WinOldApp</code> specified as the <code>wParam</code> parameter in the <code>WM_DDE_REQUEST</code> message.
message	Specifies the <code>WM_DDE_DATA</code> message.
wParam	Specifies the handle of the shell's DDE window.
LOWORD(IParam)	Specifies a handle to a global memory object that contains a <code>DDEDATA</code> structure. A description of the contents of the <code>DDEDATA</code> structure follows this table. To report an error, the server should use one of the error values listed with the <code>WinExec</code> function.
HIWORD(IParam)	Specifies an atom that represents the string, "GetIcon".

The low-order word of the `IParam` parameter is a handle to a global memory object that contains icon-property data. The icon data should be in the following form:

```
typedef struct tagICONPROPS { /* ip */
    unsigned reserved:12, /* reserved*/
    fResponse:1, /* always 1*/
    fRelease:1, /* 1 if app. frees object, else 0 */
    reserved:1, /* reserved*/
    fAckReq:1; /* 1 if app. should respond, else 0 */
    int cfFormat; /* clipboard format (not used) */
    int nWidth; /* width, in pixels, of the icon */
    int nHeight; /* height, in pixels, of the icon */
    BYTE nPlanes; /* number of planes in XOR mask*/
};
```

```
    BYTE nBitsPerPixel;    /* number of bits/pixel in XOR mask */
    LPBYTE lpANDbits; /* points to AND mask array */
    LPBYTE lpXORbits; /* points to XOR mask array */
} ICONPROPS;
```

If the server sets the `fAckReq` bit, `WinOldApp` responds to the `WM_DDE_DATA` message with a `WM_DDE_ACK` message after processing the data.

If the server sets the `fRelease` bit, `WinOldApp` frees the global memory object after copying the working-directory string. Otherwise, `WinOldApp` does not free the memory object.

The `lpANDbits` and `lpXORbits` pointers may be either near or far. If the pointers are near (that is, the segment selector portion of the pointers is zero), the bits are part of the global memory object. The offset portion of the pointers is a near offset from byte zero of the object.

Because the bits are part of the global memory object, they are freed along with the object. The combined size of the `ICONPROPS` structure together with the bits pointed to by the `lpANDbits` and `lpXORbits` members must be no more than 64K.

If the server needs to use far pointers for the `lpANDbits` and `lpXORbits` members, the bits must be part of a separate memory object. This object is not freed automatically when the global memory object is freed.

#### Terminating the DDE Conversation

The shell may terminate the conversation at any time by posting a `WM_DDE_TERMINATE` message. After `WinOldApp` has obtained its properties from the shell, it terminates the DDE conversation by posting a `WM_DDE_TERMINATE` message.

## International Applications

The Microsoft Windows operating system provides means for making applications country- and language-independent. This topic describes how to design Windows applications so that they can be readily adapted to international markets. The following topics are related to the information in this topic:

- File version library
- Resources and Resource Compiler (RC)
- Initialization files

### Creating an International Application

To reach worldwide audiences, you need to design Windows applications so that they can be marketed in more than one country and modified for new markets. International applications must be country- and language-independent and easy to localize.

A Windows application, regardless of the language used in its interface, should be able to handle data from different countries and in different languages. For example, a database developed primarily for the English-speaking market should accept French and German input. The application should also support different currency symbols and date and time formats. Furthermore, it should permit complex operations, such as sorting, in any language selected by the user.

A Windows application should be developed so that localization can be easily accomplished. Localization is the process of adapting an application for a market other than the one for which it was originally designed. Adapting an application involves translating the product, adding new features when required, and modifying the product to meet local needs.

### Achieving Country and Language Independence

Windows provides resources for writing applications that are country- and language-independent. These resources consist of international information stored in the WIN.INI file and in certain Windows functions. By using the resources described in this section, you can correctly produce international applications.

### International Information in WIN.INI

The [Intl] section of the WIN.INI file contains the current country settings for Windows. The user can modify these settings through Control Panel. An application has access to the current country settings through the `GetProfileInt` and `GetProfileString` functions and can modify them through the `WriteProfileString` function. An application should read the required country settings at startup and should monitor the `WM_WININICHANGE` message to update its country settings in case the country settings in WIN.INI have changed.

Following are the country settings stored in WIN.INI:

iCountry	Country code. This value is based on the telephone country code. The only exception is Canada, which has 2 instead of 1 (1 is used by the United States). This setting controls country-dependent features not supported by Windows.
sCountry	String defining the selected country name.
sLanguage	National language code selected by the user. The International dialog box in Control Panel changes the language of the installed language-dependent module. Following are some of the language codes that Windows currently supports:
Code	Language
DAN	Danish
DEU	German
ENG	U.K. English
ENU	U.S. English
ESN	Modern Spanish
ESP	Castilian Spanish
FIN	Finnish
FRA	French
FRC	Canadian French
ISL	Icelandic
ITA	Italian
NLD	Dutch
NOR	Norwegian



	PTG	Portuguese
	SVE	Swedish
sList	List separator. This character separates elements in a list. The list separator must be different from the decimal separator to avoid conflicts with lists of numbers.	
iMeasure	Measurement system selected by the user, where 0 equals metric and 1 equals English. This setting controls measurement-dependent features of an application.	
iTime	Time format. This setting defines the time format: 12 hours or 24 hours, where 0 equals the 12-hour clock and 1 equals the 24-hour clock.	
sTime	Time separator. This character is displayed between hours and minutes and between minutes and seconds.	
s1159	Trailing string (A.M., for example) used in some countries for times between 00:00 and 11:59.	
s2359	Trailing string (P.M., for example) for times between 12:00 and 23:59 when in 12-hour clock format or trailing string (GMT, for example) for any time when in 24-hour clock format.	
iTLZero	Value specifying whether the hours displayed should have a leading zero, where 0 equals no leading zero (9:15, for example) and 1 equals a leading zero (09:15, for example).	
iDate	Date format. Kept for compatibility with Windows 2.x. The values for this setting are: 0 equals Month-Day-Year, 1 equals Day-Month-Year, and 2 equals Year-Month-Day. The sShortDate setting should be used instead.	
sDate	Date separator. Kept for compatibility with Windows 2.x. The sShortDate setting should be used instead.	
sShortDate	Date picture of the short date format. The sShortDate setting accepts only the values m, mm, d, dd, yy and yyyy. For information about these values and the format of date pictures, see the sLongDate setting.	
sLongDate	Date picture of the long date format, which is similar to the sShortDate setting, except it can also contain strings. Following are formats for different month (m), day (d), and year (y) values:	
	Value	Format
	m	1-12
	mm	01-12
	mmm	Jan-Dec
	mmmm	January-December
	d	1-31
	dd	01-31
	ddd	Mon-Sun
	dddd	Monday-Sunday
	yy	00-99
	yyyy	1900-2040
	Following are examples of different date pictures:	
	Date picture	Example
	d mmmm, yyyy	9 January, 1989
	dddd, mmmm d, yyyy	Friday, February 7, 1992
	m/d/yy	3/18/89
	dd-mm-yyyy	18-03-1989
	d "of" mmmm, yyyy	9 of January, 1992
sCurrency	Currency symbol of a given country. Use of this setting requires care. If the currency symbol is changed through Control Panel, do not make global replacements of currency amounts in your application. Once the user has entered an amount using a particular currency, that currency should stay the same. This setting also requires special attention when files are shared among users or applications.	
iCurrency	Currency format. The values for this setting are as follows:	
	Value	Meaning
	0	Currency symbol prefix with no separation (\$1, for example)
	1	Currency symbol suffix with no separation (1\$, for example)
	2	Currency symbol prefix with one character separation (\$ 1, for example)

	3	Currency symbol suffix with one character separation (1 \$, for example)
iCurrDigits		Number of digits used for the fractional part of a currency amount.
iNegCurr		Negative currency format. The values for this setting are:
	Value	Negative format
	0	(\$1)
	1	-\$1
	2	\$-1
	3	\$1-
	4	(1\$)
	5	-1\$
	6	1- \$
	7	1\$-
	8	-1 \$
	9	-\$ 1
	10	\$ 1-
		Note: The dollar symbol represents any currency symbol defined by the sCurrency setting.
sThousand		Symbol used to separate thousands in numbers with more than three digits.
sDecimal		Character used to separate the integer part from the fractional part of a number.
iDigits		Value defining the number of decimal digits that should be used in a number.
iLzero		Value specifying whether a decimal value less than 1.0 (and greater than -1.0) should contain a leading zero, as follows:
	Value	Meaning
	0	Do not use a leading zero (.7, for example).
	1	Use a leading zero (0.7, for example).

#### International Information in Windows Functions

Windows includes provisions for specifying a national language. Language, in conjunction with the specification of a country, allows Windows to describe more precisely the characteristics of a given geographical location (for example, Swiss-German as opposed to Swiss-French). The following Windows functions behave differently depending on the language that is selected:

<u>AnsiLower</u>	<u>IsCharAlpha</u>
<u>AnsiLowerBuff</u>	<u>IsCharAlphaNumeric</u>
<u>AnsiNext</u>	<u>IsCharLower</u>
<u>AnsiPrev</u>	<u>IsCharUpper</u>
<u>AnsiUpper</u>	<u>Istrcmp</u>
<u>AnsiUpperBuff</u>	<u>Istrcmpi</u>

#### Comparing and Sorting Strings

The Istrcmp and Istrcmpi functions allow applications to compare and sort strings based on the language specified by the user. These functions take into account different alphabetic orderings, diacritical marks, and special cases that require character compression or expansion. Note that the Istrcmp and Istrcmpi functions do not act the same way as the C run-time functions strcmp and strcmpi.

The comparison done by Istrcmp and Istrcmpi is based on a primary value and a secondary value (see the following illustration). Each character has a primary and a secondary value.

When performing the comparison of two strings, the primary value takes precedence over the secondary value. That is, the secondary value is ignored unless a comparison based on primary value shows the strings as equivalent.

The following examples show the effect of primary and secondary values on string comparisons:

Comparison	Result
A = A	Primary values equal
A < a	Primary values equal, secondary values unequal (A < a)
Ab < ab	Primary values equal, secondary values unequal (A < a)
ab < Ac	Primary values unequal (b < c)

The `lstricmp` function ignores the effect of case in determining secondary value. That is, when `lstricmp` is called to compare AB and ab, the two strings are equivalent. However, `lstricmp` does not ignore diacritical marks, so Ab precedes äb regardless of whether the comparison is performed by the `lstrcmp` or `lstricmp` function.

When strings of different lengths are compared, length takes precedence over secondary values. That is, the shorter string always precedes the longer string as long as the primary values in the shorter string equal the primary values for equivalent characters in the longer string. For example, ab precedes ABC, but ABC precedes AD.

Depending on the language module installed, some characters are treated differently. For example, if the German language module is installed, the ß character expands to ss. If the Spanish language module is installed, the characters ch are treated as a single character that sorts between c and d.

#### Case Conversions

Use of the case conversion functions, `AnsiLower`, `AnsiLowerBuff`, `AnsiUpper`, and `AnsiUpperBuff`, varies depending on the language module installed. The `IsCharAlpha`, `IsCharAlphaNumeric`, `IsCharLower` and `IsCharUpper` functions are also language-dependent. Different languages treat case conversions differently.

Note: Do not use the C-language case-conversion functions; they do not handle characters with values greater than 128 properly.

#### Handling Character Sets

If you are writing international Windows applications, you will handle different character sets. It is especially important in this case to understand the difference between the Windows and OEM character sets.

The Windows character set is essentially equivalent to the ANSI character set.

The OEM character set is defined by the Windows operating system as the character set used by MS-DOS. The term OEM does not refer to a specific character set; instead, it refers to any of the different character sets (code pages) that can be installed and used by MS-DOS.

Because Windows runs on top of MS-DOS, there must be a layer between Windows and MS-DOS that performs translations between Windows and OEM characters. When Windows is first installed, the Windows Setup program looks at the character set that has been installed by MS-DOS and then installs the correct translation tables and Windows OEM fonts.

Windows applications should use the Windows `AnsiToOem` and `OemToAnsi` functions when transferring information to and from MS-DOS. Also, applications should use the correct character set when creating filenames. For more information about handling filenames, see the following section.

There is no one-to-one mapping between the Windows and OEM character sets. Applying the `AnsiToOem` function and then the `OemToAnsi` function to a given string does not always result in the original string.

Because the Windows and OEM character sets are 8-bit character sets, always use unsigned char values instead of signed char values. Bugs that result from using signed char values are very hard to track.

#### Handling Filenames

Applications do file handling differently depending on factors such as speed, size, and programming style. This section describes the most common methods for handling filenames.

The easiest way of handling filenames in Windows is to use the Windows character set for all filenames and to use the `_lcreat`, `_lopen`, and `OpenFile` functions to deal with differences between the MS-DOS and the OEM character sets.

Another way to handle filenames is to use the `OpenFile` function to obtain a full path, by using the `szPathName` member from the `OFSTRUCT` structure. The `szPathName` member contains characters from the OEM character set and must first be converted to the Windows character set before it is used as a parameter for the `OpenFile` function, for other Windows functions, or in a dialog box.

The following example shows this conversion:

```
if (OpenFile("myfile.txt", &of, OF_EXISTS) == -1) {
    OemToAnsi(of.szPathName, szAnsiPath);
    OpenFile(szAnsiPath, &of, OF_CREATE);
}
```

The third, and maybe most complicated, way of handling files is to call MS-DOS directly (by using the DOS3Call function or an Interrupt 21h instruction). You must ensure that your application always passes OEM characters to MS-DOS.

Differences between the Windows and OEM character sets complicate the handling of filenames. Problems can occur when applications try to create filenames using the Windows character set that have no equivalent characters in the OEM set. For example, the character Ê does not exist in code page 437 (437 is the standard U.S. extended ASCII character set). If the application tries to save the file named Ê.TXT, Windows converts Ê.TXT into E.TXT (by using the AnsiToOem function) and then passes it to MS-DOS.

You can prevent confusion about filenames by using the ES\_OEMCONVERT and CBS\_OEMCONVERT control styles. These styles (the first for edit controls and the second for combo boxes) read the user's input and convert the typed character to a valid character (one that exists in the OEM character set). This way, the user sees on the screen the actual filename that will be stored at the MS-DOS level.

### Handling the Keyboard

The most important keyboard issue for international applications is the use of the VK\_OEM keys for user input because the locations of these keys change depending on the keyboard layout chosen by the user.

The VkKeyScan function is used to translate characters from the Windows character set into a virtual-key code plus a shift state. This function can be also used when one application has to send text to another application by simulating keyboard input.

Some other useful keyboard functions are the following:

Function	Purpose
<u>ToAscii</u>	Converts a virtual-key code plus a shift state to a character in the Windows character set. This function is the opposite of the <u>VkKeyScan</u> function.
<u>GetKeyNameText</u>	Retrieves a string that contains the name of a key (the SHIFT key or the ENTER key, for example). The string is in the language associated with the keyboard. For example, for a French keyboard layout the names of the keys are in French.
<u>GetKBCodePage</u>	Returns the code page (OEM character set) that was running at the MS-DOS level at the time Windows was installed. Note that there is no real relationship between the keyboard and the code page installed.

To type characters that are not on your keyboard, use the ALT key and the numeric keypad. For characters in the Windows character set, hold down ALT and then, using the numeric keypad, type 0 and the three-digit code of the character you want. For an OEM character, type the three-digit code for the character.

### Handling Initialization Files

The WIN.INI and SYSTEM.INI files use the Windows character set. Usually, however, applications do not access SYSTEM.INI. For WIN.INI as well as for private initialization files, applications should use the following functions:

<u>GetPrivateProfileInt</u>	<u>GetProfileString</u>
<u>GetPrivateProfileString</u>	<u>WritePrivateProfileString</u>
<u>GetProfileInt</u>	<u>WriteProfileString</u>

The Windows character set should always be used with these functions.

The section names and setting names in WIN.INI and in private initialization files should be independent of the language of the application. Usually, all of these names remain in English. For example, in WIN.INI the section name [Desktop] and the setting name Wallpaper should always remain in English so that applications in different languages can access the same information.

### International Uses of the File Version Library

If your application includes a Windows version resource, you can use the functions in the file version library (VER.DLL) in your installation program. A Windows version resource includes the language, code page, version number, and so on for a file. The functions in VER.DLL retrieve information from a file's version resource and install files based on this information. For example, if an installation program tries to replace an existing copy of an application with a new copy in a different language, the VerInstallFile function returns an error that indicates a language conflict. Then the installation program queries the user about whether to overwrite the old file, install the new copy in another location, or exit.

For more information about the contents of a version resource and about using version functions, see File Installation Overview.

## Achieving Easy Localization

Creating applications that are easy to localize is not difficult if you follow a few basic rules.

### Isolation of Localizable Information

The most important rule for localization is to never mix functional code with strings, messages, or any other information that has to be modified to localize your application.

Hard-coded strings (strings mixed with functional code) make localization more difficult. In most Windows applications, all menus, strings and messages should be placed in the resource-definition (.RC) file. All the dialog box information should be placed in the dialog box script (.DLG) file. If you do this, you just need to run the Resource Compiler (RC) to obtain a new, localized version of the product instead of recompiling the executable file.

Strings that are not meant to be modified (filenames, WIN.INI setting names, and so on) can be placed in the .RC file, but the file should contain comments documenting that the names are permanent and should not be modified. It is a good idea also to mark what should be translated (explaining limitations, if any). The better you make the documentation, the easier the localization will be.

The .RC files and .DLG files should contain anything that can be a localization item. It is better to have extra information in these files than to have too little. In cases where an .RC or a .DLG file cannot be used, place all the information in a file, such as an include file, that is separate from any functional code.

### Allocating Extra Space for Strings

Many languages are more verbose than English and require more space to hold strings or to display dialog boxes. There are cases, as with menus, where the space allocation is done dynamically, but in most cases the application has to provide the space. The following table shows the percentage of additional space that an application should allocate for non-English strings of various lengths.

Length in characters	Additional space required
1-10	200%
11-20	100%
21-30	80%
31-50	60%
51-70	40%
70+	30%

In the English version of your application avoid creating dense menus where most of the available space (all except one line, for example) is used. Dialog boxes should be designed so that items can be moved freely, allowing reorganization of the contents as translation demands. Do not crowd status bars with information. Even abbreviations are often longer in other languages.

### Handling Foreign Languages

Never make assumptions about language usage when dealing with foreign languages. The ordering of words can be different, and the number of words required is often greater than in English.

Keep in mind the following grammatical points when preparing an application for localizing:

- Avoid using the same word in more than one message. Some words, such as none, can have different translations (different gender and number) depending on the context.
- Do not create plurals of words by adding s. Keep two strings, one for the singular and one for the plural.
- Avoid using slang, abbreviations, or jargon, because they are difficult to translate.

Keep also these syntactical considerations in mind when localizing:

- Avoid parsing text to obtain information. Parsing normally assumes specific syntax.
- Do not create a long string from several short strings. The long string may not make sense in another language, because the order of parts of speech varies in different languages.

Incorporate graphic objects such as bitmaps, cursors, and icons with these considerations in mind:

- Avoid the use of embedded text in graphics. Text is difficult to modify when in graphical form. If you cannot avoid this, leave enough space for translation and try to create tools to simplify the modification.
- Look for graphic objects that represent international concepts, because graphic objects are also language dependent.

Keep in mind the following points when planning screen elements:

- Do not hard-code the position or size of any element on the screen, because an item changes position and size as it gets translated. In cases where you need to define the size or position of certain object, place this definition in the resource-definition (.RC) file.
- Use the `CreateWindow` function carefully. The `lpClassName` parameter should be constant and independent from localization, but the `lpWindowName` parameter, which is the string that appears in the title bar, should be localized. The string used for `lpWindowName` should be taken from the resources.

All messages should be self-contained, not dynamically assembled. In cases where messages have variables added to them at run time, do not make any assumptions about the position of the variable in the message. Handle variables in messages in the following manner:

- 1 Place the string containing the variable in the resource-definition (.RC) file:

```
CannotOpen, "The application could not open the file %s"
```

- 2 Use the `wsprintf` function to incorporate the variable into the string:

```
LoadString(hInst, CannotOpen, lpFormat, MaxLen);  
wsprintf(FinalString, lpFormat, FileName);
```

## Developing Network Applications

As local area networks (LANs) become increasingly common, application developers need to ensure that their applications run properly in a network environment. To do this, they should consider the behavior of applications shared by multiple users and the compatibility of applications that access network software directly with protected (standard or 386 enhanced) mode.

### Sharing by Multiple Users

Many corporations choose to have their computer users share a single copy of an application that resides on a network server. The Microsoft Windows operating system, version 3.0 and later, can be run this way. The /n (network) option used in Windows Setup configures the user's system so that most Windows files are used directly off the network, but the user's personal files and configuration information are stored in a private Windows directory.

If you intend to allow shared copies of your application, you must ensure that two users running the same application do not interfere with each other. The following sections present guidelines for preparing an application for network support.

### Sharing Directories

Many applications store configuration files in the same directory as the executable file for the application. This method does not work for multiple users, however, because the application stores each user's information in the same directory, overwriting the other users' information in the process.

Instead of using configuration files, an application should use the Windows profile functions to store user-specific information in initialization (.INI) files. The profile functions create initialization files in a user's private Windows directory, unless the application specifies a different directory.

Windows profile functions, such as WriteProfileString, usually store profile and configuration information in .INI files. Profile functions fall into two categories: those that access WIN.INI and those that access another .INI file specified by the program.

The functions that access WIN.INI are GetProfileString, GetProfileInt, and WriteProfileString. Because each user has a unique copy of WIN.INI, these functions can be used safely, even when the application is being shared by more than one user.

The functions that access other .INI files are GetPrivateProfileString, GetPrivateProfileInt, and WritePrivateProfileString. These functions behave similarly to the functions that access WIN.INI, except that the application specifies the name of the private initialization file. When using these functions, you should specify the name of the file, but not a complete path (for example, MYAPP.INI instead of C:\MYAPP\MYAPP.INI). By default, the file will be located in the user's private Windows directory; specifying a full path could give multiple users access to the same file.

The exception to the preceding rule are initialization files that need to be shared by all users. Make sure that those files cannot be left in an inconsistent state if multiple users update them simultaneously.

### Sharing Temporary Storage

When creating temporary files, use the GetTempFileName function to determine a unique name and location for the file. This function ensures that temporary filenames do not conflict, even if multiple users share the same temporary storage directory.

### Sharing Files

A network manages file sharing as if the SHARE utility were loaded. Each file that can be accessed on the network should use a sharing mode to ensure data integrity. Applications should also be designed to handle sharing violations.

A sharing violation occurs when one process (or machine) attempts to access a file after a different process has requested the server to block access to the file. If an application opens the file in compatibility mode, a sharing violation results in a critical error. Therefore, unless the application uses the SetErrorMode function to set the error mode so that it always fails, Windows displays the standard sharing violation message.

### Sharing Devices

Windows 3.1 includes three functions that an application can use to manage its network connections: WNetAddConnection, WNetCancelConnection, and WNetGetConnection. The WNetAddConnection function redirects a local device (either a disk drive or a printer port) to a shared device on a remote server. The WNetCancelConnection function cancels a redirection to a shared device. The

WNetGetConnection function returns the name of the network resource associated with a redirected local device.

#### Calling Network Software in Protected Mode

Windows applications running in protected mode require special support whenever they make a call to real-mode software. This includes calls to MS-DOS, the BIOS, or a network. Non-Windows applications running with Windows do not require this special support, however, because they always run in real or virtual-8086 mode.

Windows applications running in protected mode require application programming interface (API) mapping. If the arguments to the calling function include pointers to data, that data should be copied into the first 1 megabyte of address space so that the real-mode software can access it. The processor is then switched into real or virtual-8086 mode so that the real-mode software can process the function. Finally, when the function returns, any data it modified is copied back to the caller's protected-mode address.

Fortunately, most applications interact with the network only indirectly, by using MS-DOS functions to manipulate files on redirected drives or by using MS-DOS or BIOS functions to print to a remote printer using redirected printer ports. Windows applications can continue to perform these functions as usual, because Windows automatically maps standard MS-DOS and BIOS functions.

Some applications, however, need to use functions that are specific to a particular network or networking protocol. Some part of the software must map these functions, and, in some cases, this may require special procedures on the part of the programmer.

The remainder of this topic describes programming considerations for designing Windows applications that use the following networking protocols and networks: Microsoft Networks and MS-DOS network functions, NetBIOS functions, Microsoft LAN Manager–based networks, Novell NetWare, Ungermann-Bass Net/One, and Banyan VINES.

#### Microsoft Networks and MS-DOS Network Functions

Many networks on the market today are based on the Microsoft Networks standard, also known as MS-NET. These networks support a set of standard MS-DOS functions that perform network activities, such as redirecting drive letters.

Current versions of Windows automatically handle these MS-DOS functions. However, in order to maintain compatibility with future Windows products, your application should not make MS-DOS calls by using Interrupt 21h. Instead, it should set up all the registers for Interrupt 21h and then make a far call to the Windows DOS3Call function.

#### NetBIOS Functions

NetBIOS is the most widely used networking API. The functions in this API are normally called by using Interrupt 5Ch. Current versions of Windows handle most NetBIOS functions. However, in order to maintain compatibility with future Windows products, the application should not make the NetBIOS call by using Interrupt 5Ch. Instead, it should set up all the registers for Interrupt 5Ch and then make a far call to the Windows NetBIOSCall function.

Windows does not support the following rarely used NetBIOS functions:

Function number	Function name
71h	Send.No.Ack
72h	Chain.Send.No.Ack
73h	Lan.Status.Alert
78h	Find.Name
79h	Trace

#### LAN Manager Networks

Networks based on Microsoft LAN Manager can be installed in either basic or enhanced versions. All versions of LAN Manager support MS-NET and NetBIOS functions. However, if you are running the enhanced version of LAN Manager with the API option, your applications can also use a powerful set of networking functions.

Non-Windows applications can call networking functions by linking with DOSNET.LIB, a static-link library provided with the network software. Windows applications, however, must use two dynamic-link libraries (DLLs), NETAPI.DLL and PMSPL.DLL, distributed on every workstation with the enhanced version of LAN Manager 2.0. (These DLLs do not run with LAN Manager 1.x or with the basic version of LAN Manager 2.0.)



For more details on writing Windows applications for LAN Manager, see the Microsoft LAN Manager Programmer's Reference.

#### Novell NetWare

Novell NetWare supports MS-NET and, optionally, NetBIOS functions, which are described earlier in this topic. Novell NetWare also supports the NetWare and IPX/SPX APIs, both based on Interrupt 21h.

Windows applications cannot make NetWare calls by using Interrupt 21h directly, because this method is not supported in all Windows operating modes. Instead, the Interrupt 21h instruction should be replaced by a far call to the NetWareRequest function. This function is exported by name from the NetWare DLL and should be imported to the module-definition (.DEF) file as NetWare.  
NetWareRequest.

Windows applications cannot make IPX/SPX calls at this time, although Novell plans to make this support available in a future release. For more information, contact Novell product support.

#### Ungermann-Bass Net/One

Ungermann-Bass Net/One is based on the Microsoft Networks standard. It supports standard MS-NET functions and most NetBIOS functions described earlier in this topic.

Net/One also supports private extensions to the NetBIOS function set (Interrupt 5Ch Functions 72h– 7Dh). These functions are supported by Windows. You can call these functions as you would standard NetBIOS functions by making a far call to the NetBIOSCall function.

#### Banyan VINES

Banyan VINES supports the standard MS-NET functions and, optionally, NetBIOS functions. A toolkit is available for applications that write directly to the VINES API.

Windows applications can call the MS-NET and NetBIOS functions as previously described.

VINES version 4.0 does not support Windows applications that call the VINES API directly, but Banyan intends to make this support available in VINES 4.1. For more information, contact Banyan product support.

## Windows Applications with MS-DOS Functions

This topic describes the support in the Microsoft Windows operating system version 3.0 and later for Windows and non-Windows applications using DOS Protected-Mode Interface (DPMI) version 1.0 functions, MS-DOS interrupts and functions in protected mode, and the NetBIOS in protected mode.

DPMI enables MS-DOS applications to access the extended memory of PC-architecture computers while maintaining system protection. It also defines a new interface, through Interrupt 31h, that protected-mode applications use for such tasks as allocating memory, modifying descriptors, and calling real-mode software.

According to the DPMI specification, the term real-mode software refers to code that runs in the low 1-megabyte address space and uses segment:offset addressing. With Windows 3.0 and later in protected mode, so-called real-mode software is actually run in virtual-8086 mode. However, because virtual-8086 mode is a close approximation of real mode, both are referred to as real mode in this topic.

For more information about the DPMI specification, contact Intel Corporation product support, or submit a service request through Microsoft OnLine.

### Using DOS Protected-Mode Interface Functions

Windows 3.0 and later in 386 enhanced mode supports DPMI version 1.0. Windows 3.0 and later in standard mode supports a subset of DPMI that enables applications to call terminate-and-stay-resident (TSR) programs and device drivers running in real (or virtual-8086) mode. To ease the porting of an application to other operating environments, all code that calls DPMI functions directly should reside in a dynamic-link library (DLL).

### Windows Kernel

Windows applications should not use the MS-DOS memory management functions for DPMI. The Windows 3.0 and later kernel has two functions, GlobalDOSAlloc and GlobalDOSFree, that should be used by Windows applications and DLLs for allocating and freeing MS-DOS addressable memory.

Because the Windows kernel provides functions for allocating memory, manipulating descriptors, and locking memory, no DPMI functions other than the following are required for Windows applications:

Interrupt 21h function	Description
0200h	Get Real Mode Interrupt Vector
0201h	Set Real Mode Interrupt Vector
0300h	Simulate Real Mode Interrupt
0301h	Call Real Mode Procedure with Far Return Frame
0302h	Call Real Mode Procedure with Interrupt Return Frame
0303h	Allocate Real Mode Callback Address
0304h	Free Real Mode Callback Address

Non-Windows applications running in 386 enhanced mode can use all DPMI functions, because those functions are not restricted by the kernel.

### Other Application Programming Interfaces

In general, any software-interrupt function that passes parameters in the EAX, EBX, ECX, EDX, ESI, EDI, and EBP registers works as long as none of the registers contains a selector value. In other words, if a software-interrupt function is completely register-based without any pointers, segment registers, or stack parameters, that function should work with Windows running in protected mode.

More complex software interrupt functions require the calling function to use the DPMI translation functions.

### Support for MS-DOS Interrupts

This section discusses support for MS-DOS interrupts and functions when Windows runs in protected mode with MS-DOS version 3.0 and later.

All MS-DOS interrupts and functions that are not mentioned in this section should work exactly as documented in *The MS-DOS Encyclopedia* (Redmond, Washington: Microsoft Press, 1988).

### Unsupported MS-DOS Interrupts and Functions

The following MS-DOS interrupts are not supported in protected mode and will fail if called:

Interrupt	Description
-----------	-------------

20h	Terminate Program
25h	Absolute Disk Read
26h	Absolute Disk Write
27h	Terminate and Stay Resident

The following MS-DOS Interrupt 21h functions are also not supported in protected mode:

Function	Description
00h	Terminate Process
0Fh	Open File with FCB
10h	Close File with FCB
14h	Sequential Read
15h	Sequential Write
16h	Create File with FCB
21h	Random Read
22h	Random Write
23h	Get File Size
24h	Set Random Record Number
27h	Random Block Read
28h	Random Block Write

#### Partially Supported MS-DOS Interrupt 21h Functions

The following MS-DOS Interrupt 21h functions behave differently in protected mode than they do in real mode. To use these functions, an application might require additional code:

Function	Description
25h	Set Interrupt Vector
35h	Get Interrupt Vector
38h	Get/Set Current Country Information
4402-4405h	Send/Receive Control Data
440Ch	Generic IOCTL for Character Devices
6501-6506h	Get Extended Country Information

Functions 25h and 35h set and get the protected-mode interrupt vector. They can be used to hook hardware interrupts, such as the timer or keyboard interrupt, as well as to hook software interrupts. (Except for Interrupts 23h, 24h, and 1Ch, software interrupts that are issued in real mode are not passed to protected-mode interrupt handlers. However, all hardware interrupts are passed to protected-mode interrupt handlers before being passed to real mode).

Function 38h returns a 34-byte buffer containing a doubleword real-mode address. The address at offset 12h is used for case mapping. To call the case-mapping function, use the DPMI translation function to simulate a real-mode FAR call.

Functions 4402h, 4403h, 4404h, and 4405h are used to receive data from a device or send data to a device. Because it is not possible to break the transfers automatically into small pieces, the calling program should assume that a transfer of greater than 4K will fail unless the address of the buffer is in the low 1 megabyte.

Only certain extensions of Function 440Ch (Minor Codes 45h and 65h) are supported for protected mode. The extensions of Function 440Ch that are used for code-page switching (Minor Codes 4Ah, 4Ch, 4Dh, 6Ah, and 6Bh) are not supported for protected-mode programs. To use 440Ch to switch code pages, you must use the DPMI translation functions.

Functions 6501h, 6502h, 6503h, 6504h, 6505h, and 6506h are supported for protected-mode programs. However, all doubleword parameters returned will contain real-mode addresses (that is, the case-conversion procedure address and all the pointers to tables will contain real-mode segment:offset addresses). To call the case-conversion procedure in real mode, you must use the DPMI translation functions.

#### NetBIOS Support

Windows supports standard NetBIOS (Interrupt 5Ch) functions in protected mode. All network control blocks (NCBs) and buffers must reside in fixed memory that is page-locked. To ease the porting of the application to other operating systems, all code that calls NetBIOS functions directly should reside in a DLL.

For more information about developing applications for networks, see [Developing Network Applications](#).

## Windows Prologs and Epilogs

This topic describes the prolog and epilog used with far functions in applications and dynamic-link libraries (DLLs) for the Microsoft Windows operating system. Compiler vendors can use this information to enable their compilers to generate prolog and epilog code that is suitable for Windows.

In Windows version 3.0 and earlier, the prolog and epilog for far functions must include instructions to mark the stack frame, indicating that the frame belongs to a far function. This makes it possible for real-mode Windows to locate segment addresses on the stack and update those addresses when it moves or discards the corresponding segments. Marking stack frames for far functions also allows debugging applications, such as Microsoft CodeView® for Windows (CVW) and Microsoft Windows 80386 Debugger (WDEB386.EXE), to display meaningful information about the contents of an application's stack.

Marking stack frames for far functions is optional for Windows 3.1 applications. Old debugging applications that do not access TOOLHELP.DLL, however, still need marking. Debugging applications that use TOOLHELP.DLL do not require stack frames for far functions to be marked.

### Data-Segment Initialization

The Windows prolog and epilog contain instructions that initialize the DS register, setting the register to the segment address of the application or DLL. Windows requires callback functions, such as window, dialog box, and enumeration procedures, to initialize the DS register whenever they are called by Windows or an application. This guarantees that the function accesses its own data segment rather than the data segment of the caller.

### Exported Far Functions

The Windows prolog used with exported far functions, such as dialog box and enumeration procedures, ensures that the DS register receives the data segment address for the application when Windows or an application calls the exported function. In Windows version 3.0 and earlier, the prolog and epilog for exported far functions have the following form:

```
push    ds; put DS in AX, take 3 bytes to do it,
popax;  so the code can be rewritten as
nop    ;   MOV AX, IMM when appropriate

incbp;  push odd BP to indicate this stack
push    bp;   frame corresponds to a far CALL

movbp, sp ; set up BP to access arguments and
;   local variables

push    ds; save DS
movds, ax ; set DS to proper data segment
subsp, const ; allocate local storage (optional)

...

subbp, 2 ; restore registers
movsp, bp
popds
popbp
decbp
retf
```

Because Windows 3.1 does not support real mode, the inc bp and dec bp instructions are not required. Also, a variety of other changes can be made to the prolog and epilog to improve speed and reduce the size of the code. If a far function is part of an application (not part of a DLL), the SS register is already the proper value for the DS register, so calling the MakeProcInstance function is not necessary. The prolog and epilog can be modified as follows:

```
push    bp; set up stack frame (optional)
movbp, sp

push    ds; save calling function's DS
```

```

push    ss; move SS to DS
popds

...

popds; restore registers
popbp

retf

```

An alternative form of the prolog and epilog for far functions follows:

```

push    bp; set up stack frame (optional)
movbp, sp

push    ds; save calling function's DS

movax, ss ; move SS to DS
movds, ax

subsp, const ; (optional) allocate local storage

...

movds, [bp-2] ; restore registers
leave

retf

```

Each of the variations of prolog and epilog code discussed previously works whether or not a far function is exported. The code can be called by an application or DLL as well as by the system.

If an application copies the contents of the SS register to the DS register, it doesn't need to call the `MakeProcInstance` function to obtain a procedure-instance address before calling an exported far function. Similarly, if a DLL moves the DGROUP data segment to the DS register through the AX register, the DLL doesn't need to call `MakeProcInstance` before calling an exported far function.

Although window procedures for an application require this same prolog, Windows loads the AX register before calling these procedures. An application, therefore, never needs to create a procedure-instance address for its window procedures.

#### Nonexported Far Functions

Although not required, nonexported far functions can also include prolog code that initializes the DS register. In this case, it is assumed that the function is never called by Windows or an application and that the DS register contains the correct segment address when the function is called. The prolog for a nonexported function has the following form:

```

movax, ds ; copy DS to AX
nop

push    bp; set up stack frame (optional)
movbp, sp

push    ds; save calling function's DS
movds, ax ; move same value back to DS

...

popds; pop same value back to DS
popbp
retf

```

An alternative form of the prolog for a nonexported function follows:

```

push    ds; copy DS to AX
popax
nop

```

```

push    bp; set up stack frame (optional)
movbp, sp

push    ds; save calling function's DS
movds, ax ; move same value back to DS

...

popds; pop same value back to DS
popbp
retf

```

A compiler should not generate the preceding code by default because it reloads the DS register with the same value two times per far call. Loading segment registers is a slow operation in protected mode and should be avoided as much as possible.

#### Exported Far Functions in a Dynamic-Link Library

Exported far functions in DLLs also require a prolog. The prolog code in a DLL must generate a reference to the DGROUP data segment. The SS register cannot be used because execution occurs on the calling function's stack. Exported far functions cannot use this method because fixups to DGROUP are illegal for a multiple instance application.

The prolog and epilog for exported far functions in a DLL has the following form:

```

movax, DGROUP ; get DGROUP value

push    bp      ; set up stack frame (optional)
movbp, sp

push    ds      ; save calling function's DS
movds, ax; move DGROUP to DS

...

popds      ; restore registers
popbp

retf

```

Following is an alternative form of the prolog for exported far functions in a DLL:

```

movax, DGROUP ; get DGROUP value

push    bp      ; set up stack frame (optional)
movbp, sp

push    ds      ; save calling function's DS
movds, ax; move DGROUP to DS

subsp, const  ; allocate local storage (optional)

...

movds, [bp-2] ; restore registers

leave

```

Windows inserts the current data segment address as the second operand (DGROUP) of the initial mov instruction.

#### Prologs in Real Mode

When Windows 3.0 and earlier is running in real mode, Windows must walk each application stack whenever it moves or discards segments. In particular, it must check each stack for any segment addresses that may have been affected by the segment operations.

To help Windows locate segment addresses associated with the stack frames of far functions, the Windows prolog increments the old frame pointer, contained in the BP register, before saving it on the stack. Because all stack offsets, including frame pointers, are expected to be word-aligned, incrementing the BP register gives Windows a quick way to locate all far function stack frames.

Windows only walks the stack in real mode. In protected mode, selector values do not change even though Windows may move and discard segments. Therefore, functions in protected mode do not need to increment the BP register when they save it. However, some debugging programs, such as CVW and WDEB386.EXE, use the incremented BP register to determine which stack frames correspond to far functions and give meaningless stack backtraces if the BP register is not incremented before it is saved.

#### Prologs in Protected Mode

Although exported functions in protected-mode, single-instance applications need to set the DS register, these functions do not require the exported prolog described in the previous section. Instead, they can use code similar to that generated by the `_loadds` keyword of the Microsoft C Optimizing Compiler (CL) to set the DS register.

The code generated by `_loadds` copies the data segment selector to the DS register whenever the function is called. Because a selector does not change value when the corresponding segment is moved, there is no need to set the AX register to the appropriate data segment address before calling the function (or to mark the stack frame). The function can, therefore, be called directly rather than through a procedure-instance address. The `_loadds` code has the following form:

```
push    bp
movbp, sp
push    ds
movax, CONSTANT
movds, ax
```

Functions that use the `_loadds` code can be used as callback functions. Because no prolog code is required, the functions do not need to be exported when used in an application. Functions in DLLs can also use the `_loadds` code. However, the functions must be exported to ensure that other applications can link dynamically to them.

In multiple-instance applications, the Windows prolog is needed only for far functions called by Windows. For these functions, procedure-instance addresses are required. The `_loadds` code cannot be used in multiple-instance applications. Instead, applications should copy the SS register to the DS register.



## Windows Application Startup

This topic describes the startup requirements of applications for the Microsoft Windows operating system. It also discusses the steps needed to initialize an application before its entry-point function, WinMain, can be called.

### Startup Requirements

When Windows starts an application, it calls a startup routine supplied with the application rather than the application's WinMain function. The startup routine is responsible for initializing the application, calling WinMain, and exiting the application when WinMain returns control.

When Windows first calls the startup routine, the processor registers have the following values:

Register	Value
AX	Contains zero.
BX	Specifies the size, in bytes, of the stack.
CX	Specifies the size, in bytes, of the heap.
DI	Contains a handle identifying the new application instance.
SI	Contains a handle identifying the previous application instance.
BP	Contains zero.
ES	Contains the segment address of the program segment prefix (PSP).
DS	Contains the segment address of the automatic data segment for the application.
SS	Same as the DS register.
SP	Contains the offset to the first byte of the application stack.

To initialize and exit a Windows application, the startup routine must follow these steps:

- 1 Initialize the task by using the InitTask function. InitTask also returns values that the startup routine passes to the WinMain function.
- 2 Clear the event that started the task by calling the WaitEvent function.
- 3 Initialize the queue and support routines for the application by calling the InitApp function with the instance handle returned by the InitTask function.
- 4 Call the entry point for the application, the WinMain function.
- 5 Exit the application by calling the MS-DOS End Program function (Interrupt 21h Function 4Ch) when WinMain returns.

Although the startup routine is essentially the same for all Windows applications, a variety of startup routines may need to be developed to accommodate the different memory models and high-level language run-time libraries used by Windows applications. If a Windows application uses functions and variables provided by run-time libraries, the startup routine may need to be customized to initialize the library at the same time as the application. Customizing the startup routine for run-time library initialization is entirely dependent on the library and is, therefore, beyond the scope of this topic.

### Example of a Startup Routine

A startup routine initializes and exits a Windows application. The routine in the following example, the \_\_astart function, shows the code needed for startup, which includes Cmacros defined in the CMACROS.INC header file. When assembled, this code is suitable for small-model Windows applications that do not use run-time libraries:

```
.xlist
memS = 1      ; small memory model
?DF = 1; Do not generate default segment definitions.
?PLM = 1;
?WIN = 1;
include cmacros.inc
.list

STACKSLOP = 256

createSeg  __TEXT, CODE, PARA, PUBLIC, CODE
createSeg  NULL, NULL, PARA, PUBLIC, BEGDATA, DGROUP
createSeg  _DATA, DATA, PARA, PUBLIC, DATA, DGROUP
defGrp  DGROUP, DATA
```

```

assumes DS,DATA

sBegin NULL
    DD 0
labelW <PUBLIC,rsrvptrs>
maxRsrvPtrs = 5
    DW maxRsrvPtrs
    DW maxRsrvPtrs DUP (0)
sEnd NULL

sBegin DATA
staticW hPrev,0 ; Save WinMain parameters.
staticW hInstance,0
staticD lpszCmdline,0
staticW cmdShow,0
sEnd DATA

externFP <INITTASK>
externFP <WAITEVENT>
externFP <INITAPP>
externFP <DOS3CALL>
externP <WINMAIN>

sBegin CODE
assumes CS,CODE

labelNP <PUBLIC,__astart>

    xorbp,bp ; zero bp
    push bp

    cCall INITTASK ; Initialize the task.
    or ax,ax
    jz noint

    addcx,STACKSLOP ; Add in stack slop space.
    jc noint ; If overflow, return error.

    movhPrev,si
    movhInstance,di
    movword ptr lpszCmdline,bx
    movword ptr lpszCmdline+2,es
    movcmdShow,dx

    xorax,ax ; Clear initial event that
    cCall WAITEVENT,<ax>; started this task.
    cCall INITAPP,<hInstance>; Initialize the queue.
    or ax,ax
    jz noint

    cCall WINMAIN,<hInstance,hPrev,lpszCmdline,cmdShow>
ix:
    movah,4Ch
    cCall DOS3CALL ; Exit with return code from app.
noint:
    moval,0FFh ; Exit with error code.
    jmp short ix
sEndCODE

    end __astart; start address

```

Windows requires the null segment (containing the rsrvptrs array), which is defined at the beginning of this sample. The InitTask function copies the top, minimum, and bottom address offsets of the stack into

the third, fourth, and fifth elements of the `rsrvptrs` array. Applications can use these offsets to check the amount of space available on the stack. The debugging version of Windows also uses these offsets to check the stack. Applications must, therefore, not change these offsets, since doing so can cause a system debugging error (RIP).

## Self-Loading Windows Applications Overview (3.1)

This topic describes the contents of a unique segment that is found only in self-loading applications for the Microsoft Windows operating system. This segment contains six functions: three that the application developer supplies and three that the Windows kernel supplies. The segment also contains a table of pointers to these functions and loader code.

This topic contains references to the Windows (new-style) header and the data tables in a Windows executable file.

### Loader Functions

The Windows kernel provides a loader function that places applications into memory and passes execution to a specified entry point. Some Windows applications, however, must bypass this kernel function and load themselves in order to be executed correctly. For example, a compiler for Windows might contain two floating-point modules: one requiring a math coprocessor and one emulating the coprocessor. The standard loader function in the Windows kernel does not provide a method of specifying that code in one module should be loaded in place of code in another; this means that the compiler needs to load the appropriate code itself in order to run efficiently and correctly. Likewise, the code for a Windows application might be compressed with a special compression algorithm in order to fit on a certain number of disks, but the standard loader function does not provide a method for dealing with a compressed file format. The application, therefore, must load itself in order to be executed correctly.

To indicate that a Windows application is self-loading, the 16-bit flag value in the executable file's Windows header must contain the value 0x0800 (that is, bit 11 must be set). Otherwise, Windows ignores the private loader code and installs the application by using the standard loader functions in the Windows kernel.

### Loader Data Table

In addition to the loader functions, the first segment of a self-loading Windows application contains a loader data table with far pointers to each of the loader functions. The format of this table follows:

Location	Description
0x00	Specifies the version number (this value must be 0xA0).
0x02	Reserved.
0x04	Points to a startup procedure, which the application developer provides.
0x08	Points to a reloading procedure, which the application developer provides.
0x0C	Reserved.
0x10	Points to a memory-allocation procedure, which the kernel provides.
0x14	Points to an entry-number procedure, which the kernel provides.
0x18	Points to an exit procedure, which the application developer provides.
0x1C	Reserved.
0x1E	Reserved.
0x20	Reserved.
0x22	Reserved.
0x24	Points to a set-owner procedure, which the kernel provides.

All of the pointers in this table must point to locations within the first segment. There can be no fixups outside this segment.

After the segment table for an executable file is loaded into memory, each entry contains an additional 16-bit value. This value is a segment selector (or handle) that the loader created.

### Loader Code

The first segment of a self-loading Windows application contains loader code for the six required loader functions. The code loads and reloads segments and resets hardware.

### Loading Segments

The kernel calls the BootApp function supplied by the application developer, instead of loading the application in the normal manner, if the 16-bit value in the information block for the Windows header contains the value 0x0800 (that is, bit 11 is set). The BootApp function allocates memory for all segments by calling the kernel-supplied MyAlloc function. If the segment is identified as a PRELOAD or FIXED type, BootApp also calls the LoadAppSeg function (another function supplied by the

application developer). The BootApp function also calls SetOwner, a kernel-supplied function, to associate the correct information block with each segment handle.

The first segment that the BootApp function should allocate is the application's automatic data segment. This data segment contains the application's stack. The automatic data segment must be allocated before the BootApp function calls the Windows PatchCodeHandle function. For more information about the PatchCodeHandle function, see the Microsoft Windows Programmer's Reference, Volume 2.

#### Reloading Segments

In addition to loading segments, the LoadAppSeg function reloads segments that the Windows kernel has discarded. Because the LoadAppSeg function is responsible for reloading segments, it must update bits 1 and 2 of the 16-bit flag value in the segment table. (Only self-loading applications should alter the Windows header or the data tables that follow it.) Bit 1 specifies whether memory is allocated for the segment, and bit 2 specifies whether the segment is currently loaded. For a complete description of the segment table, see Executable-File Format.

If the loader allocates memory for a segment but the segment is not loaded (that is, bit 1 is set and bit 2 is not), the LoadAppSeg function should call the Windows GlobalHandle function to determine whether memory is allocated for the segment. If memory is not allocated, the LoadAppSeg function should call the Windows GlobalReAlloc function to reallocate memory for the segment.

Once memory is allocated, the LoadAppSeg function should read the segment from the executable file and call the PatchCodeHandle function to correct each function prolog that occurs in the segment. Once the function prologs are altered, the LoadAppSeg function should resolve any far pointers that occur in the segment. If the pointer is specified by an ordinal value, the LoadAppSeg function should call the kernel-supplied EntryAddrProc function to resolve the address.

#### Resetting Hardware

When closing a self-loading application, the kernel calls the ExitProc function, supplied by the application developer, to reset any hardware that a dynamic-link library may have accessed. However, the ExitProc function does not need to free memory or close files.

#### Function Reference

This section provides information about the functions supplied by the application developer and by the kernel for self-loading Windows applications.

See Also

BootApp, EntryAddrProc, ExitProc, MyAlloc, PatchCodeHandle, LoadAppSeg, SetOwner

## Graphics File Formats

This topic describes the graphics-file formats used by the Microsoft Windows operating system. Graphics files include bitmap files, icon-resource files, and cursor-resource files.

### Bitmap-File Formats

Windows bitmap files are stored in a device-independent bitmap (DIB) format that allows Windows to display the bitmap on any type of display device. The term "device independent" means that the bitmap specifies pixel color in a form independent of the method used by a display to represent color. The default filename extension of a Windows DIB file is .BMP.

### Bitmap-File Structures

Each bitmap file contains a bitmap-file header, a bitmap-information header, a color table, and an array of bytes that defines the bitmap bits. The file has the following form:

```
BITMAPFILEHEADER bmfh;  
BITMAPINFOHEADER bmih;  
RGBQUADaColors[];  
BYTE aBitmapBits[];
```

The bitmap-file header contains information about the type, size, and layout of a device-independent bitmap file. The header is defined as a BITMAPFILEHEADER structure.

The bitmap-information header, defined as a BITMAPINFOHEADER structure, specifies the dimensions, compression type, and color format for the bitmap.

The color table, defined as an array of RGBQUAD structures, contains as many elements as there are colors in the bitmap. The color table is not present for bitmaps with 24 color bits because each pixel is represented by 24-bit red-green-blue (RGB) values in the actual bitmap data area. The colors in the table should appear in order of importance. This helps a display driver render a bitmap on a device that cannot display as many colors as there are in the bitmap. If the DIB is in Windows version 3.0 or later format, the driver can use the biClrImportant member of the BITMAPINFOHEADER structure to determine which colors are important.

The BITMAPINFO structure can be used to represent a combined bitmap-information header and color table.

The bitmap bits, immediately following the color table, consist of an array of BYTE values representing consecutive rows, or "scan lines," of the bitmap. Each scan line consists of consecutive bytes representing the pixels in the scan line, in left-to-right order. The number of bytes representing a scan line depends on the color format and the width, in pixels, of the bitmap. If necessary, a scan line must be zero-padded to end on a 32-bit boundary. However, segment boundaries can appear anywhere in the bitmap. The scan lines in the bitmap are stored from bottom up. This means that the first byte in the array represents the pixels in the lower-left corner of the bitmap and the last byte represents the pixels in the upper-right corner.

The biBitCount member of the BITMAPINFOHEADER structure determines the number of bits that define each pixel and the maximum number of colors in the bitmap. These members can have any of the following values:

Value	Meaning
1	Bitmap is monochrome and the color table contains two entries. Each bit in the bitmap array represents a pixel. If the bit is clear, the pixel is displayed with the color of the first entry in the color table. If the bit is set, the pixel has the color of the second entry in the table.
4	Bitmap has a maximum of 16 colors. Each pixel in the bitmap is represented by a 4-bit index into the color table. For example, if the first byte in the bitmap is 0x1F, the byte represents two pixels. The first pixel contains the color in the second table entry, and the second pixel contains the color in the sixteenth table entry.
8	Bitmap has a maximum of 256 colors. Each pixel in the bitmap is represented by a 1-byte index into the color table. For example, if the first byte in the bitmap is 0x1F, the first pixel has the color of the thirty-second table entry.
24	Bitmap has a maximum of 2 <sup>24</sup> colors. The <u>bmiColors</u> (or <u>bmciColors</u> ) member is <u>NULL</u> , and each 3-byte sequence in the bitmap array represents the relative intensities of red, green, and blue, respectively, for a pixel.

The biClrUsed member of the BITMAPINFOHEADER structure specifies the number of color indexes in

the color table actually used by the bitmap. If the `biClrUsed` member is set to zero, the bitmap uses the maximum number of colors corresponding to the value of the `biBitCount` member.

An alternative form of bitmap file uses the `BITMAPCOREINFO`, `BITMAPCOREHEADER`, and `RGBTRIPLE` structures.

### Bitmap Compression

Windows versions 3.0 and later support run-length encoded (RLE) formats for compressing bitmaps that use 4 bits per pixel and 8 bits per pixel. Compression reduces the disk and memory storage required for a bitmap.

#### Compression of 8-Bits-per-Pixel Bitmaps

When the `biCompression` member of the `BITMAPINFOHEADER` structure is set to `BI_RLE8`, the DIB is compressed using a run-length encoded format for a 256-color bitmap. This format uses two modes: encoded mode and absolute mode. Both modes can occur anywhere throughout a single bitmap.

#### Encoded Mode

A unit of information in encoded mode consists of two bytes. The first byte specifies the number of consecutive pixels to be drawn using the color index contained in the second byte.

The first byte of the pair can be set to zero to indicate an escape that denotes the end of a line, the end of the bitmap, or a delta. The interpretation of the escape depends on the value of the second byte of the pair, which must be in the range `0x00` through `0x02`. Following are the meanings of the escape values that can be used in the second byte:

Second byte	Meaning
0	End of line.
1	End of bitmap.
2	Delta. The two bytes following the escape contain unsigned values indicating the horizontal and vertical offsets of the next pixel from the current position.

#### Absolute Mode

Absolute mode is signaled by the first byte in the pair being set to zero and the second byte to a value between `0x03` and `0xFF`. The second byte represents the number of bytes that follow, each of which contains the color index of a single pixel. Each run must be aligned on a word boundary.

Following is an example of an 8-bit RLE bitmap (the two-digit hexadecimal values in the second column represent a color index for a single pixel):

Compressed data	Expanded data
03 04	04 04 04
05 06	06 06 06 06 06
00 03 45 56 67 00	45 56 67
02 78	78 78
00 02 05 01	Move 5 right and 1 down
02 78	78 78
00 00	End of line
09 1E	1E 1E 1E 1E 1E 1E 1E 1E 1E
00 01	End of RLE bitmap

#### Compression of 4-Bits-per-Pixel Bitmaps

When the `biCompression` member of the `BITMAPINFOHEADER` structure is set to `BI_RLE4`, the DIB is compressed using a run-length encoded format for a 16-color bitmap. This format uses two modes: encoded mode and absolute mode.

#### Encoded Mode

A unit of information in encoded mode consists of two bytes. The first byte of the pair contains the number of pixels to be drawn using the color indexes in the second byte.

The second byte contains two color indexes, one in its high-order nibble (that is, its low-order 4 bits) and one in its low-order nibble. The first pixel is drawn using the color specified by the high-order nibble, the second is drawn using the color in the low-order nibble, the third is drawn with the color in the high-order nibble, and so on, until all the pixels specified by the first byte have been drawn.

The first byte of the pair can be set to zero to indicate an escape that denotes the end of a line, the end of the bitmap, or a delta. The interpretation of the escape depends on the value of the second byte of the pair. In encoded mode, the second byte has a value in the range 0x00 through 0x02. The meaning of these values is the same as for a DIB with 8 bits per pixel.

#### Absolute Mode

In absolute mode, the first byte contains zero, the second byte contains the number of color indexes that follow, and subsequent bytes contain color indexes in their high- and low-order nibbles, one color index for each pixel. Each run must be aligned on a word boundary.

Following is an example of a 4-bit RLE bitmap (the one-digit hexadecimal values in the second column represent a color index for a single pixel):

Compressed data	Expanded data
03 04	0 4 0
05 06	0 6 0 6 0
00 06 45 56 67 00	4 5 5 6 6 7
04 78	7 8 7 8
00 02 05 01	Move 5 right and 1 down
04 78	7 8 7 8
00 00	End of line
09 1E	1 E 1 E 1 E 1 E 1
00 01	End of RLE bitmap

#### Bitmap Example

The following example is a text dump of a 16-color bitmap (4 bits per pixel):

```
Win3DIBFile
  BitmapFileHeader
    Type 19778
    Size 3118
    Reserved1 0
    Reserved2 0
    OffsetBits 118
  BitmapInfoHeader
    Size 40
    Width 80
    Height 75
    Planes 1
    BitCount 4
    Compression 0
    SizeImage 3000
    XPelsPerMeter 0
    YPelsPerMeter 0
    ColorsUsed 16
    ColorsImportant 16
  Win3ColorTable
    Blue Green Red Unused
[00000000] 84 252 252 84 0
[00000001] 252 252 252 84 0
[00000002] 84 84252 0
[00000003] 252 84252 0
[00000004] 84 252 252 0
[00000005] 252 252 252 0
[00000006] 00 0 0
[00000007] 168 0 0 0
[00000008] 0168 0 0
[00000009] 168 168 0 0
[0000000A] 00 168 0
[0000000B] 168 0 168 0
[0000000C] 0168 168 0
[0000000D] 168 168 168 0
[0000000E] 84 8484 0
```



```
[0000000F] 252 8484 0
Image
.
.   Bitmap data
.
```

### Icon-Resource File Format

An icon-resource file contains image data for icons used by Windows applications. The file consists of an icon directory identifying the number and types of icon images in the file, plus one or more icon images. The default filename extension for an icon-resource file is .ICO.

### Icon Directory

Each icon-resource file starts with an icon directory. The icon directory, defined as an ICONDIR structure, specifies the number of icons in the resource and the dimensions and color format of each icon image. The ICONDIR structure has the following form:

```
typedef struct ICONDIR {
    WORD idReserved;
    WORD idType;
    WORD idCount;
    ICONDIRENTRY idEntries[1];
} ICONHEADER;
```

Following are the members in the ICONDIR structure:

idReserved	Reserved; must be zero.
idType	Specifies the resource type. This member is set to 1.
idCount	Specifies the number of entries in the directory.
idEntries	Specifies an array of ICONDIRENTRY structures containing information about individual icons. The idCount member specifies the number of structures in the array.

The ICONDIRENTRY structure specifies the dimensions and color format for an icon. The structure has the following form:

```
struct IconDirectoryEntry {
    BYTE bWidth;
    BYTE bHeight;
    BYTE bColorCount;
    BYTE bReserved;
    WORD wPlanes;
    WORD wBitCount;
    DWORD dwBytesInRes;
    DWORD dwImageOffset;
};
```

Following are the members in the ICONDIRENTRY structure:

bWidth	Specifies the width of the icon, in pixels. Acceptable values are 16, 32, and 64.
bHeight	Specifies the height of the icon, in pixels. Acceptable values are 16, 32, and 64.
bColorCount	Specifies the number of colors in the icon. Acceptable values are 2, 8, and 16.
bReserved	Reserved; must be zero.
wPlanes	Specifies the number of color planes in the icon bitmap.
wBitCount	Specifies the number of bits in the icon bitmap.
dwBytesInRes	Specifies the size of the resource, in bytes.
dwImageOffset	Specifies the offset, in bytes, from the beginning of the file to the icon image.

### Icon Image

Each icon-resource file contains one icon image for each image identified in the icon directory. An icon image consists of an icon-image header, a color table, an XOR mask, and an AND mask. The icon image has the following form:

```
BITMAPINFOHEADER icHeader;
RGBQUAD icColors[];
BYTE icXOR[];
```

```
BYTE icAND[];
```

The icon-image header, defined as a `BITMAPINFOHEADER` structure, specifies the dimensions and color format of the icon bitmap. Only the `biSize` through `biBitCount` members and the `biSizeImage` member are used. All other members (such as `biCompression` and `biClrImportant`) must be set to zero.

The color table, defined as an array of `RGBQUAD` structures, specifies the colors used in the XOR mask. As with the color table in a bitmap file, the `biBitCount` member in the icon-image header determines the number of elements in the array. For more information about the color table, see Section 1.1, "Bitmap-File Formats."

The XOR mask, immediately following the color table, is an array of `BYTE` values representing consecutive rows of a bitmap. The bitmap defines the basic shape and color of the icon image. As with the bitmap bits in a bitmap file, the bitmap data in an icon-resource file is organized in scan lines, with each byte representing one or more pixels, as defined by the color format. For more information about these bitmap bits, see Section 1.1, "Bitmap-File Formats."

The AND mask, immediately following the XOR mask, is an array of `BYTE` values, representing a monochrome bitmap with the same width and height as the XOR mask. The array is organized in scan lines, with each byte representing 8 pixels.

When Windows draws an icon, it uses the AND and XOR masks to combine the icon image with the pixels already on the display surface. Windows first applies the AND mask by using a bitwise AND operation; this preserves or removes existing pixel color. Windows then applies the XOR mask by using a bitwise XOR operation. This sets the final color for each pixel.

The following illustration shows the XOR and AND masks that create a monochrome icon (measuring 8 pixels by 8 pixels) in the form of an uppercase K:

#### Windows Icon Selection

Windows detects the resolution of the current display and matches it against the width and height specified for each version of the icon image. If Windows determines that there is an exact match between an icon image and the current device, it uses the matching image. Otherwise, it selects the closest match and stretches the image to the proper size.

If an icon-resource file contains more than one image for a particular resolution, Windows uses the icon image that most closely matches the color capabilities of the current display. If no image matches the device capabilities exactly, Windows selects the image that has the greatest number of colors without exceeding the number of display colors. If all images exceed the color capabilities of the current display, Windows uses the icon image with the least number of colors.

#### Cursor-Resource File Format

A cursor-resource file contains image data for cursors used by Windows applications. The file consists of a cursor directory identifying the number and types of cursor images in the file, plus one or more cursor images. The default filename extension for a cursor-resource file is `.CUR`.

#### Cursor Directory

Each cursor-resource file starts with a cursor directory. The cursor directory, defined as a `CURSORDIR` structure, specifies the number of cursors in the file and the dimensions and color format of each cursor image. The `CURSORDIR` structure has the following form:

```
typedef struct _CURSORDIR {
    WORD cdReserved;
    WORD cdType;
    WORD cdCount;
    CURSORDIRENTRY cdEntries[];
} CURSORDIR;
```

Following are the members in the `CURSORDIR` structure:

<code>cdReserved</code>	Reserved; must be zero.
<code>cdType</code>	Specifies the resource type. This member must be set to 2.
<code>cdCount</code>	Specifies the number of cursors in the file.
<code>cdEntries</code>	Specifies an array of <code>CURSORDIRENTRY</code> structures containing information about individual cursors. The <code>cdCount</code> member specifies the number of structures in the array.

A `CURSORDIRENTRY` structure specifies the dimensions and color format of a cursor image. The structure has the following form:

```
typedef struct _CURSORDIRENTRY {
    BYTE  bWidth;
    BYTE  bHeight;
    BYTE  bColorCount;
    BYTE  bReserved;
    WORD  wXHotspot;
    WORD  wYHotspot;
    DWORD lBytesInRes;
    DWORD dwImageOffset;
} CURSORDIRENTRY;
```

Following are the members in the `CURSORDIRENTRY` structure:

<code>bWidth</code>	Specifies the width of the cursor, in pixels.
<code>bHeight</code>	Specifies the height of the cursor, in pixels.
<code>bColorCount</code>	Reserved; must be zero.
<code>bReserved</code>	Reserved; must be zero.
<code>wXHotspot</code>	Specifies the x-coordinate, in pixels, of the hot spot.
<code>wYHotspot</code>	Specifies the y-coordinate, in pixels, of the hot spot.
<code>lBytesInRes</code>	Specifies the size of the resource, in bytes.
<code>dwImageOffset</code>	Specifies the offset, in bytes, from the start of the file to the cursor image.

#### Cursor Image

Each cursor-resource file contains one cursor image for each image identified in the cursor directory. A cursor image consists of a cursor-image header, a color table, an XOR mask, and an AND mask. The cursor image has the following form:

```
BITMAPINFOHEADER  crHeader;
RGBQUAD  crColors[];
BYTE  crXOR[];
BYTE  crAND[];
```

The cursor hot spot is a single pixel in the cursor bitmap that Windows uses to track the cursor. The `crXHotspot` and `crYHotspot` members specify the x- and y-coordinates of the cursor hot spot. These coordinates are 16-bit integers.

The cursor-image header, defined as a `BITMAPINFOHEADER` structure, specifies the dimensions and color format of the cursor bitmap. Only the `biSize` through `biBitCount` members and the `biSizeImage` member are used. The `biHeight` member specifies the combined height of the XOR and AND masks for the cursor. This value is twice the height of the XOR mask. The `biPlanes` and `biBitCount` members must be 1. All other members (such as `biCompression` and `biClrImportant`) must be set to zero.

The color table, defined as an array of `RGBQUAD` structures, specifies the colors used in the XOR mask. For a cursor image, the table contains exactly two structures, since the `biBitCount` member in the cursor-image header is always 1.

The XOR mask, immediately following the color table, is an array of `BYTE` values representing consecutive rows of a bitmap. The bitmap defines the basic shape and color of the cursor image. As with the bitmap bits in a bitmap file, the bitmap data in a cursor-resource file is organized in scan lines, with each byte representing one or more pixels, as defined by the color format. For more information about these bitmap bits, see Section 1.1, "Bitmap-File Formats."

The AND mask, immediately following the XOR mask, is an array of `BYTE` values representing a monochrome bitmap with the same width and height as the XOR mask. The array is organized in scan lines, with each byte representing 8 pixels.

When Windows draws a cursor, it uses the AND and XOR masks to combine the cursor image with the pixels already on the display surface. Windows first applies the AND mask by using a bitwise AND operation; this preserves or removes existing pixel color. Window then applies the XOR mask by using a bitwise XOR operation. This sets the final color for each pixel.

The following illustration shows the XOR and the AND masks that create a cursor (measuring 8 pixels by 8 pixels) in the form of an arrow:

Following are the bit-mask values necessary to produce black, white, inverted, and transparent results:

Pixel result	AND mask	XOR mask
Black	0	0
White	0	1
Transparent	1	0
Inverted	1	1

#### Windows Cursor Selection

If a cursor-resource file contains more than one cursor image, Windows determines the best match for a particular display by examining the width and height of the cursor images.

## Clipboard File Format

Microsoft Windows Clipboard (CLIPBRD.EXE) saves and reads its data in files with the .CLP extension. A .CLP file contains a value identifying it as a Clipboard data file; one or more structures defining the format, size, and location of the data; and one or more blocks of actual data.

### Clipboard-File Header

The Clipboard data file begins with a header consisting of two members. Following are the members in this header:

FileIdentifier	Identifies the file as a Clipboard data file. This member must be set to CLP_ID. This is a 2-byte value.
FormatCount	Specifies the number of clipboard formats contained in the file. This is a 2-byte value.

### Clipboard-File Structure

The header is followed by one or more structures, each of which identifies the format, size, and offset of a block containing clipboard data. Following are the members in this structure:

FormatID	Specifies the clipboard-format identifier of the clipboard data. For a description of the various formats that are available, see the description of <u>SetClipboardData</u> . This is 2-byte value.
LenData	Specifies the length, in bytes, of the clipboard data. This is a 4-byte value.
OffData	Specifies the offset, in bytes, of the clipboard-data block. The offset is from the beginning of the file. This is a 4-byte value.
Name	Identifies a 79-character array specifying the format name of a private clipboard format.

The first block of clipboard data follows the last of these structures. For bitmaps and metafiles, the bits follow immediately after the bitmap header and the METAFILEPICT structures.

See Also

SetClipboardData, METAFILEPICT

## Metafile Format

A metafile for the Microsoft Windows operating system consists of a collection of graphics device interface (GDI) functions that describe an image. Because metafiles take up less space and are more device-independent than bitmaps, they provide convenient storage for images that appear repeatedly in an application or need to be moved from one application to another.

To generate a metafile, a Windows application creates a special device context that sends GDI commands to a file or memory for storage. The application can later play back the metafile and display the image.

During playback, Windows breaks the metafile down into records and identifies each object with an index to a handle table. When a META\_DELETEOBJECT record is encountered during playback, the associated object is deleted from the handle table. The entry is then reused by the next object that the metafile creates. To ensure compatibility, an application that explicitly manipulates records or builds its own metafile should manage the handle table in the same way. For more information on the format of the handle table, see the [HANDLETABLE](#) structure.

In some cases, there are two variants of a metafile record, one representing the record created by Windows versions before 3.0 and the second representing the record created by Windows versions 3.0 and later. Windows versions 3.0 and later play all metafile versions but store only 3.0 and later versions. Windows versions earlier than 3.0 do not play metafiles recorded by Windows versions 3.0 and later.

A metafile consists of two parts: a header and a list of records. The header and records are described in the remainder of this topic. For a list of function-specific records, see [Metafile Records](#).

### Metafile Header

The metafile header contains a description of the size of the metafile and the number of drawing objects it uses. The drawing objects can be pens, brushes, bitmaps, or fonts.

The metafile header has the following form:

```
typedef struct tagMETAHEADER {
    WORD    mtType;
    WORD    mtHeaderSize;
    WORD    mtVersion;
    DWORD   mtSize;
    WORD    mtNoObjects;
    DWORD   mtMaxRecord;
    WORD    mtNoParameters;
} METAHEADER;
```

Following are the members in the metafile header:

mtType	Specifies whether the metafile is stored in memory or recorded in a file. This member has one of the following values:
	Value      Meaning
	0            Metafile is in memory.
	1            Metafile is in a file.
mtHeaderSize	Specifies the size, in words, of the metafile header.
mtVersion	Specifies the Windows version number. The version number for Windows version 3.0 and later is 0x300.
mtSize	Specifies the size, in words, of the file.
mtNoObjects	Specifies the maximum number of objects that can exist in the metafile at the same time.
mtMaxRecord	Specifies the size, in words, of the largest record in the metafile.
mtNoParameters	Not used.

### Typical Metafile Record

The graphics device interface stores most of the GDI functions that an application can use to create metafiles in typical records.

A typical metafile record has the following form:

```
struct {
    DWORD   rdSize;
    WORD    rdFunction;
```

```

        WORD   rdParm[ ];
    }

```

Following are the members in a typical metafile record:

**rdSize** Specifies the size, in words, of the record.

**rdFunction** Specifies the function number. This value may be the number of any function in the table at the end of this section.

**rdParm** Identifies an array of words containing the function parameters (listed in the reverse order in which they are passed to the function).

Following are the GDI functions found in typical records, along with their hexadecimal values:

GDI function	Value
<u>Arc</u>	0x0817
<u>Chord</u>	0x0830
<u>Ellipse</u>	0x0418
<u>ExcludeClipRect</u>	0x0415
<u>FloodFill</u>	0x0419
<u>IntersectClipRect</u>	0x0416
<u>LineTo</u>	0x0213
<u>MoveTo</u>	0x0214
<u>OffsetClipRgn</u>	0x0220
<u>OffsetViewportOrg</u>	0x0211
<u>OffsetWindowOrg</u>	0x020F
<u>PatBlt</u>	0x061D
<u>Pie</u>	0x081A
<u>RealizePalette</u> (3.0 and later)	0x0035
<u>Rectangle</u>	0x041B
<u>ResizePalette</u> (3.0 and later)	0x0139
<u>RestoreDC</u>	0x0127
<u>RoundRect</u>	0x061C
<u>SaveDC</u>	0x001E
<u>ScaleViewportExt</u>	0x0412
<u>ScaleWindowExt</u>	0x0400
<u>SetBkColor</u>	0x0201
<u>SetBkMode</u>	0x0102
<u>SetMapMode</u>	0x0103
<u>SetMapperFlags</u>	0x0231
<u>SetPixel</u>	0x041F
<u>SetPolyFillMode</u>	0x0106
<u>SetROP2</u>	0x0104
<u>SetStretchBltMode</u>	0x0107
<u>SetTextAlign</u>	0x012E
<u>SetTextCharacterExtra</u>	0x0108
<u>SetTextColor</u>	0x0209
<u>SetTextJustification</u>	0x020A
<u>SetViewportExt</u>	0x020E
<u>SetViewportOrg</u>	0x020D
<u>SetWindowExt</u>	0x020C
<u>SetWindowOrg</u>	0x020B

#### Placeable Windows Metafiles

A placeable Windows metafile is a standard Windows metafile that has an additional 22-byte header. The header contains information about the aspect ratio and original size of the metafile, permitting applications to display the metafile in its intended form.

The header for a placeable Windows metafile has the following form:

```

typedef struct {
    DWORD    key;
    HANDLE   hmf;
    RECT     bbox;
    WORD     inch;
    DWORD    reserved;
    WORD     checksum;
} METAFILEHEADER;

```

Following are the members of a placeable metafile header:

key	Specifies the binary key that uniquely identifies this file type. This member must be set to 0x9AC6CDD7L.
hmf	Unused; must be zero.
bbox	Specifies the coordinates of the smallest rectangle that encloses the picture. The coordinates are in metafile units as defined by the inch member.
inch	Specifies the number of metafile units to the inch. To avoid numeric overflow, this value should be less than 1440. Most applications use 576 or 1000.
reserved	Unused; must be zero.
checksum	Specifies the checksum. It is the sum (using the XOR operator) of the first 10 words of the header.

The actual content of the Windows metafile immediately follows the header. The format for this content is identical to that for standard Windows metafiles. For some applications, a placeable Windows metafile must not exceed 64K.

Note: Placeable Windows metafiles are not compatible with the GetMetaFile function. Applications that intend to use the metafile functions to read and play placeable Windows metafiles must read the file by using an input function (such as \_lread), strip the 22-byte header, and create a standard Windows metafile by using the remaining bytes and the SetMetaFileBits function.

#### Guidelines for Windows Metafiles

To ensure that metafiles can be transported between different computers and applications, any application that creates a metafile should make sure the metafile is device-independent and sizable.

The following guidelines ensure that every metafile can be accepted and manipulated by other applications:

- Set a mapping mode as one of the first records. Many applications, including OLE applications, only accept metafiles that are in MM\_ANISOTROPIC mode.
- Call the SetWindowOrg and SetWindowExt functions. Do not call the SetViewportExt or SetViewportOrg functions if the user will be able to resize or change the dimensions of the object.
- Use the MFCOMMENT printer escape to add comments to the metafile.
- Rely primarily on the functions listed in Typical Metafile Record. Observe the following limitations on the functions you use:
  - Do not use functions that retrieve data (for example, GetActiveWindow or EnumFontFamilies).
  - Do not use any of the region functions (because they are device dependent).
  - Use StretchBlt or StretchDIB instead of BitBlt.

#### Sample of Metafile Program Output

This section describes a sample program and the metafile that it creates. The sample program creates a small metafile that draws a purple rectangle with a green border and writes the words "Hello People" in the rectangle.

```

MakeAMetaFile(hDC)
HDC hDC;
{
    HPENhMetaGreenPen;
    HBRUSH hMetaVioletBrush;
    HDC hDCMeta;
    HANDLE hMeta;

    /* Create the metafile with output going to the disk. */

    hDCMeta = CreateMetaFile( (LPSTR) "sample.met");

```



```

hMetaGreenPen = CreatePen(0, 0, (DWORD) 0x0000FF00);
SelectObject(hDCMeta, hMetaGreenPen);

hMetaVioletBrush = CreateSolidBrush((DWORD) 0x00FF00FF);
SelectObject(hDCMeta, hMetaVioletBrush);

Rectangle(hDCMeta, 0, 0, 150, 70);

TextOut(hDCMeta, 10, 10, (LPSTR) "Hello People", 12);

/* We are done with the metafile. */

hMeta = CloseMetaFile(hDCMeta);

/* Play the metafile that we just created. */

PlayMetaFile(hDC, hMeta);
}

```

The resulting metafile, SAMPLE.MET, consists of a metafile header and six records. It has the following binary form:

```

0001    mtType... disk metafile
0009    mtSize...
0300    mtVersion
0000 0036    mtSize
0002    mtNoObjects
0000 000C    mtMaxRecord
0000    mtNoParameters

0000 0008    rdSize
02FA    rdFunction (CreatePenIndirect function)
0000 0000 0000 0000 FF00    rdParm (LOGPEN structure defining pen)

0000 0004    rdSize
012D    rdFunction (SelectObject)
0000    rdParm (index to object #0... the above pen)

0000 0007    rdSize
02FC    rdFunction (CreateBrushIndirect)
0000 00FF 00FF 0000    rdParm (LOGBRUSH structure defining the brush)

0000 0004    rdSize
012D    rdFunction (SelectObject)
0001    rdParm (index to object #1... the brush)

0000 0007    rdSize
041B    rdFunction (Rectangle)
0046 0096 0000 0000    rdParm (parameters sent to Rectangle...
in reverse order)

0000 000C    rdSize
0521    rdFunction (TextOut)
rdParm
000C    count
string
48 65 6C 6C 6F 20 50 65 6F 70 6C 65    "Hello People"
000A    y-value
000A    x-value

```

## Metafile Records

### Function-Specific Metafile Records

The graphics-device interface stores most of the GDI functions for creating metafiles in typical records. The remainder are stored in function-specific records that contain structures in the rdParm member. This section contains definitions for these records.

AnimatePalette

BitBlt

CreateBrushIndirect

CreateFontIndirect

CreatePalette

CreatePatternBrush

CreatePenIndirect

CreateRegion

DeleteObject

Escape

ExtTextOut

Polygon

PolyPolygon

Polyline

SelectClipRgn

SelectObject

SelectPalette

SetDIBitsToDevice

SetPaletteEntries

StretchBlt

StretchDIBits

TextOut

## AnimatePalette Metafile Record

### AnimatePalette Metafile Record

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm[];  
}
```

Member	Description								
rdSize	Specifies the record size, in words.								
rdFunction	Specifies the GDI function number 0x0436.								
rdParm	Contains the following elements: <table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>start</td><td>First entry to be animated</td></tr><tr><td>numentries</td><td>Number of entries to be animated</td></tr><tr><td>entries</td><td><u>PALETTEENTRY</u> blocks.</td></tr></tbody></table>	Element	Description	start	First entry to be animated	numentries	Number of entries to be animated	entries	<u>PALETTEENTRY</u> blocks.
Element	Description								
start	First entry to be animated								
numentries	Number of entries to be animated								
entries	<u>PALETTEENTRY</u> blocks.								

## BitBlt Metafile Record

### BitBlt Metafile Record (3.0)

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm[];  
}
```

The BitBlt record contains a device-independent bitmap suitable for playback on any device.

Member	Description																				
rdSize	Specifies the record size, in words.																				
rdFunction	Specifies the GDI function number 0x0940.																				
rdParm	Contains the following elements:																				
	<table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>raster op</td><td>High-order word of the raster operation</td></tr><tr><td>SY</td><td>Y-coordinate of the source origin</td></tr><tr><td>SX</td><td>X-coordinate of the source origin</td></tr><tr><td>DYE</td><td>Destination y-extent</td></tr><tr><td>DXE</td><td>Destination x-extent</td></tr><tr><td>DY</td><td>Y-coordinate of the destination origin</td></tr><tr><td>DX</td><td>X-coordinate of the destination origin</td></tr><tr><td>BitmapInfo</td><td><u>BITMAPINFO</u> structure</td></tr><tr><td>bits</td><td>Actual device-independent bitmap bits</td></tr></tbody></table>	Element	Description	raster op	High-order word of the raster operation	SY	Y-coordinate of the source origin	SX	X-coordinate of the source origin	DYE	Destination y-extent	DXE	Destination x-extent	DY	Y-coordinate of the destination origin	DX	X-coordinate of the destination origin	BitmapInfo	<u>BITMAPINFO</u> structure	bits	Actual device-independent bitmap bits
Element	Description																				
raster op	High-order word of the raster operation																				
SY	Y-coordinate of the source origin																				
SX	X-coordinate of the source origin																				
DYE	Destination y-extent																				
DXE	Destination x-extent																				
DY	Y-coordinate of the destination origin																				
DX	X-coordinate of the destination origin																				
BitmapInfo	<u>BITMAPINFO</u> structure																				
bits	Actual device-independent bitmap bits																				

### BitBlt Metafile Record (2.x)

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm[];  
}
```

The BitBlt record stored by Windows versions earlier than 3.0 contains a device-dependent bitmap that may not be suitable for playback on all devices.

Member	Description																												
rdSize	Specifies the record size, in words.																												
rdFunction	Specifies the GDI function number 0x0922.																												
rdParm	Contains the following elements:																												
	<table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>raster op</td><td>High-order word of the raster operation</td></tr><tr><td>SY</td><td>Y-coordinate of the source origin</td></tr><tr><td>SX</td><td>X-coordinate of the source origin</td></tr><tr><td>DYE</td><td>Destination y-extent</td></tr><tr><td>DXE</td><td>Destination x-extent</td></tr><tr><td>DY</td><td>Y-coordinate of the destination origin</td></tr><tr><td>DX</td><td>X-coordinate of the destination origin</td></tr><tr><td>bmWidth</td><td>Width of bitmap, in pixels</td></tr><tr><td>bmHeight</td><td>Height of bitmap, in raster lines</td></tr><tr><td>bmWidthBytes</td><td>Number of bytes in each raster line</td></tr><tr><td>bmPlanes</td><td>Number of color planes in the bitmap</td></tr><tr><td>bmBitsPixel</td><td>Number of adjacent color bits</td></tr><tr><td>bits</td><td>Actual device-dependent bitmap bits</td></tr></tbody></table>	Element	Description	raster op	High-order word of the raster operation	SY	Y-coordinate of the source origin	SX	X-coordinate of the source origin	DYE	Destination y-extent	DXE	Destination x-extent	DY	Y-coordinate of the destination origin	DX	X-coordinate of the destination origin	bmWidth	Width of bitmap, in pixels	bmHeight	Height of bitmap, in raster lines	bmWidthBytes	Number of bytes in each raster line	bmPlanes	Number of color planes in the bitmap	bmBitsPixel	Number of adjacent color bits	bits	Actual device-dependent bitmap bits
Element	Description																												
raster op	High-order word of the raster operation																												
SY	Y-coordinate of the source origin																												
SX	X-coordinate of the source origin																												
DYE	Destination y-extent																												
DXE	Destination x-extent																												
DY	Y-coordinate of the destination origin																												
DX	X-coordinate of the destination origin																												
bmWidth	Width of bitmap, in pixels																												
bmHeight	Height of bitmap, in raster lines																												
bmWidthBytes	Number of bytes in each raster line																												
bmPlanes	Number of color planes in the bitmap																												
bmBitsPixel	Number of adjacent color bits																												
bits	Actual device-dependent bitmap bits																												

## CreateBrushIndirect Metafile Record

### CreateBrushIndirect Metafile Record

```
struct {  
    DWORD    rdSize;  
    WORDrdFunction;  
    LOGBRUSH rdParm;  
}
```

Member	Description
rdSize	Specifies the record size, in words.
rdFunction	Specifies the GDI function number 0x02FC.
rdParm	Specifies the logical brush.

## CreateFontIndirect Metafile Record

### CreateFontIndirect Metafile Record

```
struct {  
    DWORD    rdSize;  
    WORD     rdFunction;  
    LOGFONT  rdParm;  
}
```

Member	Description
rdSize	Specifies the record size, in words.
rdFunction	Specifies the GDI function number 0x02FB.
rdParm	Specifies the logical font.

## CreatePalette Metafile Record

### CreatePalette Metafile Record

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    LOGPALETTE rdParm;  
}
```

Member	Description
rdSize	Specifies the record size, in words.
rdFunction	Specifies the GDI function number 0x00F7.
rdParm	Specifies the logical palette.

## CreatePatternBrush Metafile Record

### CreatePatternBrush Metafile Record (3.0)

```
struct {  
    DWORD rdSize;  
    WORD  rdFunction;  
    WORD  rdParm[];  
}
```

The CreatePatternBrush record contains a device-independent bitmap suitable for playback on all devices.

Member	Description										
rdSize	Specifies the record size, in words.										
rdFunction	Specifies the GDI function number 0x0142.										
rdParm	Contains the following elements: <table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>type</td><td>Bitmap type. This element may be either of these two values: BS_PATTERN--Brush is defined by a device-dependent bitmap through a call to the <u>CreatePatternBrush</u> function. BS_DIBPATTERN--Brush is defined by a device-independent bitmap through a call to the <u>CreateDIBPatternBrush</u> function.</td></tr><tr><td>wUsage</td><td>Color-table type. This element specifies whether the bmiColors member of the BITMAPINFO structure contains explicit RGB values or indexes to the <u>currently realized</u> logical palette. This element must be one of the following values: DIB_RGB_COLORS--The color table contains literal <u>RGB</u> values. DIB_PAL_COLORS--The color table consists of an array of indexes to the currently realized logical palette.</td></tr><tr><td>bmi</td><td><u>BITMAPINFO</u> structure</td></tr><tr><td>bits</td><td>Actual device-independent bitmap bits.</td></tr></tbody></table>	Element	Description	type	Bitmap type. This element may be either of these two values: BS_PATTERN--Brush is defined by a device-dependent bitmap through a call to the <u>CreatePatternBrush</u> function. BS_DIBPATTERN--Brush is defined by a device-independent bitmap through a call to the <u>CreateDIBPatternBrush</u> function.	wUsage	Color-table type. This element specifies whether the bmiColors member of the BITMAPINFO structure contains explicit RGB values or indexes to the <u>currently realized</u> logical palette. This element must be one of the following values: DIB_RGB_COLORS--The color table contains literal <u>RGB</u> values. DIB_PAL_COLORS--The color table consists of an array of indexes to the currently realized logical palette.	bmi	<u>BITMAPINFO</u> structure	bits	Actual device-independent bitmap bits.
Element	Description										
type	Bitmap type. This element may be either of these two values: BS_PATTERN--Brush is defined by a device-dependent bitmap through a call to the <u>CreatePatternBrush</u> function. BS_DIBPATTERN--Brush is defined by a device-independent bitmap through a call to the <u>CreateDIBPatternBrush</u> function.										
wUsage	Color-table type. This element specifies whether the bmiColors member of the BITMAPINFO structure contains explicit RGB values or indexes to the <u>currently realized</u> logical palette. This element must be one of the following values: DIB_RGB_COLORS--The color table contains literal <u>RGB</u> values. DIB_PAL_COLORS--The color table consists of an array of indexes to the currently realized logical palette.										
bmi	<u>BITMAPINFO</u> structure										
bits	Actual device-independent bitmap bits.										

### CreatePatternBrush Metafile Record (2.x)

```
struct {  
    DWORD rdSize;  
    WORD  rdFunction;  
    WORD  rdParm[];  
}
```

The CreatePatternBrush record contains a device-dependent bitmap that may not be suitable for playback on all devices.

Member	Description																
rdSize	Specifies the record size, in words.																
rdFunction	Specifies the GDI function number 0x01F9.																
rdParm	Contains the following elements: <table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>bmWidth</td><td>Bitmap width</td></tr><tr><td>bmHeight</td><td>Bitmap height</td></tr><tr><td>bmWidthBytes</td><td>Bytes per raster line</td></tr><tr><td>bmPlanes</td><td>Number of color planes</td></tr><tr><td>bmBitsPixel</td><td>Number of adjacent color bits that define a pixel</td></tr><tr><td>bmBits</td><td>Pointer to bit values</td></tr><tr><td>bits</td><td>Actual bits of pattern</td></tr></tbody></table>	Element	Description	bmWidth	Bitmap width	bmHeight	Bitmap height	bmWidthBytes	Bytes per raster line	bmPlanes	Number of color planes	bmBitsPixel	Number of adjacent color bits that define a pixel	bmBits	Pointer to bit values	bits	Actual bits of pattern
Element	Description																
bmWidth	Bitmap width																
bmHeight	Bitmap height																
bmWidthBytes	Bytes per raster line																
bmPlanes	Number of color planes																
bmBitsPixel	Number of adjacent color bits that define a pixel																
bmBits	Pointer to bit values																
bits	Actual bits of pattern																



## CreatePenIndirect Metafile Record

### CreatePenIndirect Metafile Record

```
struct {  
    DWORD   rdSize;  
    WORD    rdFunction;  
    LOGPEN  rdParm;  
}
```

Member	Description
rdSize	Specifies the record size, in words.
rdFunction	Specifies the GDI function number 0x02FA.
rdParm	Specifies the logical pen.

## CreateRegion Metafile Record

### CreateRegion Metafile Record

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm[];  
}
```

Member	Description
rdSize	Specifies the record size, in words.
rdFunction	Specifies the GDI function number 0x06FF.
rdParm	Specifies the region to be created.

## DeleteObject Metafile Record

### DeleteObject Metafile Record

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm;  
}
```

Member	Description
rdSize	Specifies the record size, in words.
rdFunction	Specifies the GDI function number 0x01F0.
rdParm	Specifies the index to the handle table for the object to be deleted.

## Escape Metafile Record

### Escape Metafile Record

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm[];  
}
```

Member	Description								
rdSize	Specifies the record size, in words.								
rdFunction	Specifies the GDI function number 0x0626.								
rdParm	Contains the following elements: <table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>escape number</td><td>Number identifying individual escape.</td></tr><tr><td>count</td><td>Number of bytes of information.</td></tr><tr><td>input data</td><td>Variable-length field. The member is <math>((\text{count}+1) \gg 1)</math> words long.</td></tr></tbody></table>	Element	Description	escape number	Number identifying individual escape.	count	Number of bytes of information.	input data	Variable-length field. The member is $((\text{count}+1) \gg 1)$ words long.
Element	Description								
escape number	Number identifying individual escape.								
count	Number of bytes of information.								
input data	Variable-length field. The member is $((\text{count}+1) \gg 1)$ words long.								

## ExtTextOut Metafile Record

### ExtTextOut Metafile Record

```
struct{
    DWORD rdSize;
    WORD rdFunction;
    WORD rdParm[];
}
```

Member	Description																
rdSize	Specifies the record size, in words.																
rdFunction	Specifies the GDI function number 0x0A32.																
rdParm	Contains the following elements: <table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>y</td><td>Logical y-value of the starting point for the string.</td></tr><tr><td>x</td><td>Logical x-value of the starting point for the string.</td></tr><tr><td>count</td><td>Length of the string.</td></tr><tr><td>options</td><td>Rectangle type. An application should use the AND (&amp;) operator to determine if this element has either the <u>ETO_CLIPPED</u> or <u>ETO_OPAQUE</u> bits set. Using the equality operator (==) is discouraged in this case, because some applications set additional bits in the wOptions parameter of the rectangular region in which the <u>ExtTextOut</u> function writes text.</td></tr><tr><td>rectangle</td><td><u>RECT</u> structure defining the rectangular region in which the <u>ExtTextOut</u> function writes text. This element does not exist if the options element is zero.</td></tr><tr><td>string</td><td>Byte array containing the string. The array is ((count + 1) &gt;&gt; 1) words long.</td></tr><tr><td>dxarray</td><td>Optional word array of intercharacter distances.</td></tr></tbody></table>	Element	Description	y	Logical y-value of the starting point for the string.	x	Logical x-value of the starting point for the string.	count	Length of the string.	options	Rectangle type. An application should use the AND (&) operator to determine if this element has either the <u>ETO_CLIPPED</u> or <u>ETO_OPAQUE</u> bits set. Using the equality operator (==) is discouraged in this case, because some applications set additional bits in the wOptions parameter of the rectangular region in which the <u>ExtTextOut</u> function writes text.	rectangle	<u>RECT</u> structure defining the rectangular region in which the <u>ExtTextOut</u> function writes text. This element does not exist if the options element is zero.	string	Byte array containing the string. The array is ((count + 1) >> 1) words long.	dxarray	Optional word array of intercharacter distances.
Element	Description																
y	Logical y-value of the starting point for the string.																
x	Logical x-value of the starting point for the string.																
count	Length of the string.																
options	Rectangle type. An application should use the AND (&) operator to determine if this element has either the <u>ETO_CLIPPED</u> or <u>ETO_OPAQUE</u> bits set. Using the equality operator (==) is discouraged in this case, because some applications set additional bits in the wOptions parameter of the rectangular region in which the <u>ExtTextOut</u> function writes text.																
rectangle	<u>RECT</u> structure defining the rectangular region in which the <u>ExtTextOut</u> function writes text. This element does not exist if the options element is zero.																
string	Byte array containing the string. The array is ((count + 1) >> 1) words long.																
dxarray	Optional word array of intercharacter distances.																

## Polygon Metafile Record

### Polygon Metafile Record

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm[];  
}
```

Member	Description						
rdSize	Specifies the record size, in words.						
rdFunction	Specifies the GDI function number 0x0324.						
rdParm	Contains the following elements: <table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>count</td><td>Number of points</td></tr><tr><td>list of points</td><td>List of individual points</td></tr></tbody></table>	Element	Description	count	Number of points	list of points	List of individual points
Element	Description						
count	Number of points						
list of points	List of individual points						

## PolyPolygon Metafile Record

### PolyPolygon Metafile Record

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm[];  
}
```

Member	Description								
rdSize	Specifies the record size, in words.								
rdFunction	Specifies the GDI function number 0x0538.								
rdParm	Contains the following elements: <table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>count</td><td>Total number of polygons</td></tr><tr><td>list of polygon counts</td><td>List of number of points for each polygon</td></tr><tr><td>list of points</td><td>List of individual points</td></tr></tbody></table>	Element	Description	count	Total number of polygons	list of polygon counts	List of number of points for each polygon	list of points	List of individual points
Element	Description								
count	Total number of polygons								
list of polygon counts	List of number of points for each polygon								
list of points	List of individual points								

## Polyline Metafile Record

### Polyline Metafile Record

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm[];  
}
```

Member	Description						
rdSize	Specifies the record size, in words.						
rdFunction	Specifies the GDI function number 0x0325.						
rdParm	Contains the following elements: <table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>count</td><td>Number of points</td></tr><tr><td>list of points</td><td>List of individual points</td></tr></tbody></table>	Element	Description	count	Number of points	list of points	List of individual points
Element	Description						
count	Number of points						
list of points	List of individual points						



## SelectClipRgn Metafile Record

### SelectClipRgn Metafile Record

```
struct{  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm;  
}
```

Member	Description
rdSize	Specifies the record size, in words.
rdFunction	Specifies the GDI function number 0x012C.
rdParm	Specifies the index to the handle table for the region being selected.

## SelectObject Metafile Record

### SelectObject Metafile Record

```
struct{  
    DWORD rdSize;  
    WORD  rdFunction;  
    WORD  rdParm;  
}
```

Member	Description
rdSize	Specifies the record size, in words.
rdFunction	Specifies the GDI function number 0x012D.
rdParm	Specifies the index to the handle table for the object being selected.

## SelectPalette Metafile Record

### SelectPalette Metafile Record

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm;  
}
```

Member	Description
rdSize	Specifies the record size, in words.
rdFunction	Specifies the GDI function number 0x0234.
rdParm	Specifies the index to the handle table for the logical palette being selected.

## SetDIBitsToDevice Metafile Record

### SetDIBitsToDevice Metafile Record

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm[];  
}
```

Member	Description																								
rdSize	Specifies the record size, in words.																								
rdFunction	Specifies the GDI function number 0x0D33.																								
rdParm	Contains the following elements: <table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>wUsage</td><td>Flag indicating whether the bitmap color table contains <u>RGB</u> values or indexes to the currently realized logical palette</td></tr><tr><td>numscans</td><td>Number of scan lines in the bitmap</td></tr><tr><td>startscan</td><td>First scan line in the bitmap</td></tr><tr><td>srcY</td><td>Y-coordinate for the origin of the source rectangle in the bitmap</td></tr><tr><td>srcX</td><td>X-coordinate for the origin of the source rectangle in the bitmap</td></tr><tr><td>extY</td><td>Height of the source rectangle in the bitmap</td></tr><tr><td>extX</td><td>Width of the source rectangle in the bitmap</td></tr><tr><td>destY</td><td>Y-coordinate of the origin of the destination rectangle</td></tr><tr><td>destX</td><td>X-coordinate of the origin of the destination rectangle</td></tr><tr><td>BitmapInfo</td><td><u>BITMAPINFO</u> structure</td></tr><tr><td>bits</td><td>Actual device-independent bitmap bits</td></tr></tbody></table>	Element	Description	wUsage	Flag indicating whether the bitmap color table contains <u>RGB</u> values or indexes to the currently realized logical palette	numscans	Number of scan lines in the bitmap	startscan	First scan line in the bitmap	srcY	Y-coordinate for the origin of the source rectangle in the bitmap	srcX	X-coordinate for the origin of the source rectangle in the bitmap	extY	Height of the source rectangle in the bitmap	extX	Width of the source rectangle in the bitmap	destY	Y-coordinate of the origin of the destination rectangle	destX	X-coordinate of the origin of the destination rectangle	BitmapInfo	<u>BITMAPINFO</u> structure	bits	Actual device-independent bitmap bits
Element	Description																								
wUsage	Flag indicating whether the bitmap color table contains <u>RGB</u> values or indexes to the currently realized logical palette																								
numscans	Number of scan lines in the bitmap																								
startscan	First scan line in the bitmap																								
srcY	Y-coordinate for the origin of the source rectangle in the bitmap																								
srcX	X-coordinate for the origin of the source rectangle in the bitmap																								
extY	Height of the source rectangle in the bitmap																								
extX	Width of the source rectangle in the bitmap																								
destY	Y-coordinate of the origin of the destination rectangle																								
destX	X-coordinate of the origin of the destination rectangle																								
BitmapInfo	<u>BITMAPINFO</u> structure																								
bits	Actual device-independent bitmap bits																								

## SetPaletteEntries Metafile Record

### SetPaletteEntries Metafile Record

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm[ ];  
}
```

Member	Description								
rdSize	Specifies the record size, in words.								
rdFunction	Specifies the GDI function number 0x0037.								
rdParm	Contains the following elements: <table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>start</td><td>First entry to be set in the palette</td></tr><tr><td>numentries</td><td>Number of entries to be set in the palette</td></tr><tr><td>entries</td><td><u>PALETTEENTRY</u> blocks</td></tr></tbody></table>	Element	Description	start	First entry to be set in the palette	numentries	Number of entries to be set in the palette	entries	<u>PALETTEENTRY</u> blocks
Element	Description								
start	First entry to be set in the palette								
numentries	Number of entries to be set in the palette								
entries	<u>PALETTEENTRY</u> blocks								

## StretchBlt Metafile Record

### StretchBlt Metafile Record (3.0)

```
struct {
    DWORD rdSize;
    WORD rdFunction;
    WORD rdParm[];
}
```

The StretchBlt record contains a device-independent bitmap suitable for playback on all devices.

Member	Description																										
rdSize	Specifies the record size, in words.																										
rdFunction	Specifies the GDI function number 0x0B41.																										
rdParm	Contains the following elements:																										
	<table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>raster op</td><td>Low-order word of the raster operation</td></tr><tr><td>raster op</td><td>High-order word of the raster operation</td></tr><tr><td>SYE</td><td>Source y-extent</td></tr><tr><td>SXE</td><td>Source x-extent</td></tr><tr><td>SY</td><td>Y-coordinate of the source origin</td></tr><tr><td>SX</td><td>X-coordinate of the source origin</td></tr><tr><td>DYE</td><td>Destination y-extent</td></tr><tr><td>DXE</td><td>Destination x-extent</td></tr><tr><td>DY</td><td>Y-coordinate of the destination origin</td></tr><tr><td>DX</td><td>X-coordinate of the destination origin</td></tr><tr><td>BitmapInfo</td><td><u>BITMAPINFO</u> structure</td></tr><tr><td>bits</td><td>Actual device-independent bitmap bits</td></tr></tbody></table>	Element	Description	raster op	Low-order word of the raster operation	raster op	High-order word of the raster operation	SYE	Source y-extent	SXE	Source x-extent	SY	Y-coordinate of the source origin	SX	X-coordinate of the source origin	DYE	Destination y-extent	DXE	Destination x-extent	DY	Y-coordinate of the destination origin	DX	X-coordinate of the destination origin	BitmapInfo	<u>BITMAPINFO</u> structure	bits	Actual device-independent bitmap bits
Element	Description																										
raster op	Low-order word of the raster operation																										
raster op	High-order word of the raster operation																										
SYE	Source y-extent																										
SXE	Source x-extent																										
SY	Y-coordinate of the source origin																										
SX	X-coordinate of the source origin																										
DYE	Destination y-extent																										
DXE	Destination x-extent																										
DY	Y-coordinate of the destination origin																										
DX	X-coordinate of the destination origin																										
BitmapInfo	<u>BITMAPINFO</u> structure																										
bits	Actual device-independent bitmap bits																										

### StretchBlt Metafile Record (2.x)

```
struct {
    DWORD rdSize;
    WORD rdFunction;
    WORD rdParm[];
}
```

The StretchBlt record contains a device-dependent bitmap that may not be suitable for playback on all devices.

Member	Description																												
rdSize	Specifies the record size, in words.																												
rdFunction	Specifies the GDI function number 0x0B23.																												
rdParm	Contains the following elements:																												
	<table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>raster op</td><td>Low-order word of the raster operation</td></tr><tr><td>raster op</td><td>High-order word of the raster operation</td></tr><tr><td>SYE</td><td>Source y-extent</td></tr><tr><td>SXE</td><td>Source x-extent</td></tr><tr><td>SY</td><td>Y-coordinate of the source origin</td></tr><tr><td>SX</td><td>X-coordinate of the source origin</td></tr><tr><td>DYE</td><td>Destination y-extent</td></tr><tr><td>DXE</td><td>Destination x-extent</td></tr><tr><td>DY</td><td>Y-coordinate of the destination origin</td></tr><tr><td>DX</td><td>X-coordinate of the destination origin</td></tr><tr><td>bmWidth</td><td>Width of the bitmap, in pixels</td></tr><tr><td>bmHeight</td><td>Height of the bitmap, in raster lines</td></tr><tr><td>bmWidthBytes</td><td>Number of bytes in each raster line</td></tr></tbody></table>	Element	Description	raster op	Low-order word of the raster operation	raster op	High-order word of the raster operation	SYE	Source y-extent	SXE	Source x-extent	SY	Y-coordinate of the source origin	SX	X-coordinate of the source origin	DYE	Destination y-extent	DXE	Destination x-extent	DY	Y-coordinate of the destination origin	DX	X-coordinate of the destination origin	bmWidth	Width of the bitmap, in pixels	bmHeight	Height of the bitmap, in raster lines	bmWidthBytes	Number of bytes in each raster line
Element	Description																												
raster op	Low-order word of the raster operation																												
raster op	High-order word of the raster operation																												
SYE	Source y-extent																												
SXE	Source x-extent																												
SY	Y-coordinate of the source origin																												
SX	X-coordinate of the source origin																												
DYE	Destination y-extent																												
DXE	Destination x-extent																												
DY	Y-coordinate of the destination origin																												
DX	X-coordinate of the destination origin																												
bmWidth	Width of the bitmap, in pixels																												
bmHeight	Height of the bitmap, in raster lines																												
bmWidthBytes	Number of bytes in each raster line																												

bmPlanes	Number of color planes in the bitmap
bmBitsPixel	Number of adjacent color bits
bits	Actual bitmap bits

## StretchDIBits Metafile Record

### StretchDIBits Metafile Record

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm[ ];  
}
```

Member	Description																										
rdSize	Specifies the record size, in words.																										
rdFunction	Specifies the GDI function number 0x0F43.																										
rdParm	Contains the following elements: <table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>dwRop</td><td>Raster operation to be performed</td></tr><tr><td>Usag</td><td>Flag indicating whether the bitmap color table contains <u>RGB</u> values or indexes to the currently realized logical palette</td></tr><tr><td>srcYExt</td><td>Height of the source in the bitmap</td></tr><tr><td>srcXExt</td><td>Width of the source in the bitmap</td></tr><tr><td>srcY</td><td>Y-coordinate of the origin of the source in the bitmap</td></tr><tr><td>srcX</td><td>X-coordinate of the origin of the source in the bitmap</td></tr><tr><td>dstYExt</td><td>Height of the destination rectangle</td></tr><tr><td>dstXExt</td><td>Width of the destination rectangle</td></tr><tr><td>dstY</td><td>Y-coordinate of the origin of the destination rectangle</td></tr><tr><td>dstX</td><td>X-coordinate of the origin of the destination rectangle</td></tr><tr><td>BitmapInfo</td><td><u>BITMAPINFO</u> structure</td></tr><tr><td>bits</td><td>Actual device-independent bitmap bits</td></tr></tbody></table>	Element	Description	dwRop	Raster operation to be performed	Usag	Flag indicating whether the bitmap color table contains <u>RGB</u> values or indexes to the currently realized logical palette	srcYExt	Height of the source in the bitmap	srcXExt	Width of the source in the bitmap	srcY	Y-coordinate of the origin of the source in the bitmap	srcX	X-coordinate of the origin of the source in the bitmap	dstYExt	Height of the destination rectangle	dstXExt	Width of the destination rectangle	dstY	Y-coordinate of the origin of the destination rectangle	dstX	X-coordinate of the origin of the destination rectangle	BitmapInfo	<u>BITMAPINFO</u> structure	bits	Actual device-independent bitmap bits
Element	Description																										
dwRop	Raster operation to be performed																										
Usag	Flag indicating whether the bitmap color table contains <u>RGB</u> values or indexes to the currently realized logical palette																										
srcYExt	Height of the source in the bitmap																										
srcXExt	Width of the source in the bitmap																										
srcY	Y-coordinate of the origin of the source in the bitmap																										
srcX	X-coordinate of the origin of the source in the bitmap																										
dstYExt	Height of the destination rectangle																										
dstXExt	Width of the destination rectangle																										
dstY	Y-coordinate of the origin of the destination rectangle																										
dstX	X-coordinate of the origin of the destination rectangle																										
BitmapInfo	<u>BITMAPINFO</u> structure																										
bits	Actual device-independent bitmap bits																										



## TextOut Metafile Record

### TextOut Metafile Record

```
struct {  
    DWORD rdSize;  
    WORD rdFunction;  
    WORD rdParm[];  
}
```

Member	Description										
rdSize	Specifies the record size, in words.										
rdFunction	Specifies the GDI function number 0x0521.										
rdParm	Contains the following elements: <table><thead><tr><th>Element</th><th>Description</th></tr></thead><tbody><tr><td>count</td><td>Length of the string</td></tr><tr><td>string</td><td>Actual string</td></tr><tr><td>y-value</td><td>Logical y-coordinate of the starting point for the string</td></tr><tr><td>x-value</td><td>Logical x-coordinate of the starting point for the string</td></tr></tbody></table>	Element	Description	count	Length of the string	string	Actual string	y-value	Logical y-coordinate of the starting point for the string	x-value	Logical x-coordinate of the starting point for the string
Element	Description										
count	Length of the string										
string	Actual string										
y-value	Logical y-coordinate of the starting point for the string										
x-value	Logical x-coordinate of the starting point for the string										

## Font-File Format

This topic describes the file formats for raster and vector fonts used by the Microsoft Windows operating system. These file formats may be used by smart text generators in some support modules for the graphics device interface (GDI). Vector formats, however, are more frequently used by GDI than by the support modules. TrueType font files are described in TrueType Font Files, available from Microsoft Corporation.

### Organization of a Font File

Raster and vector font files begin with information that is common to both types of file and then continue with information that differs for each type. These font files are stored with an .FNT extension.

In Windows versions 3.0 and later, the font-file header for raster and vector fonts includes six new members: `dfFlags`, `dfAspace`, `dfBspace`, `dfCspace`, `dfColorPointer`, and `dfReserved1`. All device drivers support the fonts in Windows 2.x. However, not all drivers support those in versions 3.0 and later.

In Windows, font files for raster and vector fonts include the glyph table in the `dfCharTable` member, which consists of structures describing the bits for characters in the font file. The use of 32-bit offsets to the character glyphs in the `dfCharTable` member enables fonts to exceed 64K, the size limit of Windows 2.x fonts.

Because of their 32-bit offsets and potentially large size, the newer fonts are designed for use on systems that are running Windows versions 3.0 and later in protected (standard or 386-enhanced) mode and are using an 80386 (or higher) processor whose 32-bit registers can access the character glyphs. Typically, newer drivers use the newer version of a font only when both of these conditions are true.

### Font-File Structure

Font information is found at the beginning of both raster and vector font files.

Following are the members of the FONTINFO structure:

<code>dfVersion</code>	Specifies the version (0x0200 or 0x0300) of the file.
<code>dfSize</code>	Specifies the total size of the file, in bytes.
<code>dfCopyright</code>	Specifies copyright information.
<code>dfType</code>	Specifies the type of font file. This information is organized as follows: Byte                    Description Low-order               Exclusively for GDI use. If the low-order bit of the word is zero, it is a bitmap (raster) font file. If the low-order bit is 1, it is a vector font file. The second bit is reserved and must be zero. If no bits follow in the file and the bits are located in memory at a fixed address specified by the <code>dfBitsOffset</code> member, the third bit is set to 1. Otherwise, the bit is set to zero. If the font is realized by a device, the high-order bit of the low-order byte is set. The remaining bits in the low-order byte are then reserved and set to zero. High-order              Reserved for device use and is always set to zero for standard fonts realized by GDI. Physical fonts that set the high-order bit of the low-order byte may use this byte to describe themselves. GDI never inspects the high-order byte.
<code>dfPoints</code>	Specifies the nominal point size (that is, the number identifying the point size) at which this character set looks best.
<code>dfVertRes</code>	Specifies the nominal vertical resolution (that is, the number identifying the vertical resolution), in dots per inch, at which this character set was digitized.
<code>dfHorizRes</code>	Specifies the nominal horizontal resolution (that is, the number identifying the horizontal resolution), in dots per inch, at which this character set was digitized.
<code>dfAscent</code>	Specifies the distance from the top of a character-definition cell to the base line of the typographical font. The <code>dfAscent</code> member is useful for aligning the base lines of fonts with different heights.
<code>dfInternalLeading</code>	Specifies the amount of leading inside the bounds set by the <code>dfPixHeight</code> member. Accent marks can occur in this area. The designer can set the value to zero.
<code>dfExternalLeading</code>	Specifies the amount of extra leading that the designer requests the application to add between rows. Since this area is outside the font proper, it

	contains no marks and is not altered by text-output calls in either opaque or transparent mode. The designer can set the value to zero.														
dfItalic	Specifies whether the character-definition data represents an italic font. If the flag is set, the low-order bit is 1. All other bits are zero.														
dfUnderline	Specifies whether the character-definition data represents an underlined font. If the flag is set, the low-order bit is 1. All other bits are zero.														
dfStrikeOut	Specifies whether the character-definition data represents a strikeout font. If the flag is set, the low-order bit is 1. All other bits are zero.														
dfWeight	Specifies the weight of the characters in the character-definition data, on a scale of 1 through 1000. A dfWeight value of 400 specifies a regular weight.														
dfCharSet	Specifies the character set defined by this font.														
dfPixWidth	Specifies the width of the grid on which a vector font was digitized. For raster fonts, if the dfPixWidth member is nonzero, it represents the width for all the characters in the bitmap. If the member is zero, the font has variable-width characters whose widths are specified in the array for the dfCharTable member.														
dfPixHeight	Specifies the height of the character bitmap for raster fonts or the height of the grid on which a vector font was digitized.														
dfPitchAndFamily	Specifies the pitch and font family. If the font is variable pitch, the low bit is set. The four high bits give the family name of the font. Font families describe the general look of a font. They identify fonts when the exact name is not available. The font families are described as follows: <table border="0" style="margin-left: 2em;"> <tr> <td>Family</td> <td>Description</td> </tr> <tr> <td>FF_DONTCARE</td> <td>Unknown.</td> </tr> <tr> <td>FF_ROMAN</td> <td>Proportionally spaced fonts with serifs.</td> </tr> <tr> <td>FF_SWISS</td> <td>Proportionally spaced fonts without serifs.</td> </tr> <tr> <td>FF_MODERN</td> <td>Fixed-pitch fonts.</td> </tr> <tr> <td>FF_SCRIPT</td> <td>Cursive or script fonts. (Both are designed to look similar to handwriting. Script fonts have joined letters; cursive fonts do not.)</td> </tr> <tr> <td>FF_DECORATIVE</td> <td>Novelty fonts.</td> </tr> </table>	Family	Description	FF_DONTCARE	Unknown.	FF_ROMAN	Proportionally spaced fonts with serifs.	FF_SWISS	Proportionally spaced fonts without serifs.	FF_MODERN	Fixed-pitch fonts.	FF_SCRIPT	Cursive or script fonts. (Both are designed to look similar to handwriting. Script fonts have joined letters; cursive fonts do not.)	FF_DECORATIVE	Novelty fonts.
Family	Description														
FF_DONTCARE	Unknown.														
FF_ROMAN	Proportionally spaced fonts with serifs.														
FF_SWISS	Proportionally spaced fonts without serifs.														
FF_MODERN	Fixed-pitch fonts.														
FF_SCRIPT	Cursive or script fonts. (Both are designed to look similar to handwriting. Script fonts have joined letters; cursive fonts do not.)														
FF_DECORATIVE	Novelty fonts.														
dfAvgWidth	Specifies the width of characters in the font. For fixed-pitch fonts, this value is the same as the value for the dfPixWidth member. For variable-pitch fonts, it is the width of the character "X".														
dfMaxWidth	Specifies the maximum pixel width of any character in the font. For fixed-pitch fonts, this value is the same as the value of the dfPixWidth member.														
dfFirstChar	Specifies the first character code defined by the font. Character definitions are stored only for the characters actually present in the font. Use this member, therefore, when calculating indexes for either the dfBits or dfCharOffset member.														
dfLastChar	Specifies the last character code defined by the font. All characters with codes between the values for the dfFirstChar and dfLastChar members must be present in the character definitions for the font.														
dfDefaultChar	Specifies the character to substitute whenever a string contains a character that is out of range. The character is given relative to the dfFirstChar member so that the dfDefaultChar member is the actual value of the character less the value of the dfFirstChar member. The dfDefaultChar member indicates a special character that is not a space.														
dfBreakChar	Specifies the character that defines word breaks for word wrapping and word-spacing justification. The character is given relative to the dfFirstChar member so that the dfBreakChar member is the actual value of the character less that of the dfFirstChar member. The dfBreakChar member is normally 32 minus the value of the dfFirstChar member (the ASCII space character).														
dfWidthBytes	Specifies the number of bytes in each row of the bitmap. This value is always even so that the rows start on word boundaries. For vector fonts, this member has no meaning.														
dfDevice	Specifies the offset in the file to the string giving the device name. For a generic font, this value is zero.														
dfFace	Specifies the offset in the file to the null-terminated string that names the face.														

dfBitsPointer	Specifies the absolute machine address of the bitmap. This is set by GDI at load time. The value of the dfBitsPointer member is guaranteed to be even.	
dfBitsOffset	Specifies the offset in the file to the beginning of the bitmap information. If the third bit in the dfType member is set, the dfBitsOffset member is an absolute address of the bitmap (probably in read-only memory). For raster fonts, the dfBitsOffset member points to a sequence of bytes that make up the bitmap of the font. The height of the bitmap is the height of the font, and its width is the sum of the widths of the characters in the font, rounded up to the next word boundary. For vector fonts, the dfBitsOffset member points to a string of bytes or words (depending on the size of the grid on which the font was digitized) that specify the strokes for each character of the font. The value of the dfBitsOffset member must be even.	
dfReserved	Not used.	
dfFlags	Specifies the bit flags that define the format of the glyph bitmap, as follows:	
	Pitch value	Address
	DFF_FIXED	0x0001
	DFF_PROPORTIONAL	0x0002
	DFF_ABCFIXED	0x0004
	DFF_ABCPROPORTIONAL	0x0008
	DFF_1COLOR	0x0010
	DFF_16COLOR	0x0020
	DFF_256COLOR	0x0040
	DFF_RGBCOLOR	0x0080
dfAspace	Specifies the global A space, if any. The value of the dfAspace member is the distance from the current position to the left edge of the bitmap.	
dfBspace	Specifies the global B space, if any. The value of the dfBspace member is the width of the character.	
dfCspace	Specifies the global C space, if any. The value of the dfCspace member is the distance from the right edge of the bitmap to the new current position. The increment of a character is the sum of the A, B, and C spaces. These spaces apply to all glyphs, including DFF_ABCFIXED.	
dfColorPointer	Specifies the offset to the color table for color fonts, if any. The format of the bits is like a device-independent bitmap (DIB), but without the header. (That is, the characters are not split into disjoint bytes; instead, they are left intact.) If no color table is needed, this entry is NULL.	
dfReserved1	Not used.	
dfCharTable	Specifies an array of entries for raster, fixed-pitch vector, and proportionally spaced vector fonts, as follows:	
	Font type	Description
	Raster	Each entry in the array consists of two 2-byte words for Windows 2.x and three 2-byte words for Windows 3.0 and later. The first word of each entry is the character width. The second word of each entry is the byte offset from the beginning of the FONTINFO structure to the character bitmap. For Windows 3.0 and later, the second and third words are used for the offset.
	Fixed-pitch vector	Each 2-byte entry in the array specifies the offset from the start of the bitmap to the beginning of the string of stroke specification units for the character. The number of bytes or words to be used for a particular character is calculated by subtracting its entry from the next one, so that there is a sentinel at the end of the array of values.

Proportionally-spaced vector      Each 4-byte entry in the array is divided into two 2-byte fields. The first field gives the starting offset from the start of the bitmap of the character strokes. The second field gives the pixel width of the character.

One extra entry at the end of the character table describes an absolute-space character, which is guaranteed to be blank. This character is not part of the normal character set.

The number of entries in the table is calculated as follows: (dfLastChar - dfFirstChar) + 2. This number includes a "spare," the sentinel offset.

For more information on the dfCharTable member, see Section 4.3, "Version-Specific Glyph Tables."

facename      Specifies an ASCII character string that constitutes the name of the font face. The size of this member is the length of the string plus a null terminating character.

devicename      Specifies an ASCII character string that constitutes the name of the device if this font file is for a specific one. The size of this member is the length of the string plus a null terminating character.

bitmaps      Specifies character bitmap definitions. Unlike the old font format, each character is stored as a contiguous set of bytes.

The first byte contains the first 8 bits of the first scan line (that is, the top line of the character). The second byte contains the first 8 bits of the second scan line. This continues until the first "column" is completely defined. The subsequent byte contains the next 8 bits of the first scan line, padded with zeros on the right if necessary (and so on, down through the second "column"). If the glyph is quite narrow, each scan line is covered by one byte, with bits set to zero as necessary for padding. If the glyph is very wide, a third or even fourth set of bytes can be present.

Character bitmaps must be stored contiguously and arranged in ascending order. The bytes for a 12-pixel by 14-pixel "A" character, for example, are given in two sets, because the character is less than 17 pixels wide:

```
00 06 09 10 20 20 20 3F 20 20 20 00 00 00
00 00 00 80 40 40 40 C0 40 40 40 00 00 00
```

Note that in the second set of bytes, the second digit of the byte is always zero. The zeros correspond to the thirteenth through sixteenth pixels on the right side of the character, which are not used by this character bitmap.

### Version-Specific Glyph Tables

The dfCharTable member for Windows 2.x has a GlyphEntry structure with the following format:

```
GlyphEntry struc
geWidth  dw ? ; width of char bitmap, pixels
geOffset dw ? ; pointer to the bits
GlyphEntry ends
```

The dfCharTable member in Windows 3.0 and later is dependent on the format of the glyph bitmap. The only formats supported are DFF\_FIXED and DFF\_PROPORTIONAL.

```
DFF_FIXED
DFF_PROPORTIONAL
```

```
GlyphEntry struc
geWidth  dw ? ; width of char bitmap, pixels
geOffset dd ? ; pointer to the bits
GlyphEntry ends
```

```
DFF_ABCFIXED
DFF_ABCPROPORTIONAL
```

```
GlyphEntry struc
geWidth  dw ? ; width of char bitmap, pixels
```

```
geOffset dd ? ; pointer to the bits
geAspace dd ? ; A space, fract pixels (16.16)
geBspace dd ? ; B space, fract pixels (16.16)
geCspace dw ? ; C space, fract pixels (16.16)
GlyphEntry ends
```

Fractional pixels are expressed as 32-bit signed numbers with an implicit binary point between bits 15 and 16. This is referred to as a 16.16 ("sixteen dot sixteen") fixed-point number.

The ABC spacing in the following example is the same as defined previously. However, specific sets are defined for each character:

```
DFF_1COLOR; 8 pixels per byte
DFF_16COLOR ; 2 pixels per byte
DFF_256COLOR ; 1 pixel per byte
DFF_RGBCOLOR ; RGB quads
```

```
GlyphEntry struc
geWidth dw ? ; width of char bitmap, pixels
geOffset dd ? ; pointer to the bits
geHeight dw ? ; height of char bitmap, pixels
geAspace dd ? ; A space, fract pixels (16.16)
geBspace dd ? ; B space, fract pixels (16.16)
geCspace dd ? ; C space, fract pixels (16.16)
GlyphEntry ends
```

## Group File Format Overview (3.1)

This topic describes the format of group files used by the Microsoft Windows operating system. A group file contains data that Microsoft Windows Program Manager (PROGMAN.EXE) uses to display the icons of the applications in a group, start the applications in a group, and open related documents.

### Organization of a Group File

The first element in a group file is the group-file header. The data in the group-file header includes an identifier, a count of bytes, a count of items in the file, and information that the system uses to display group icons.

The group-file header is followed by one or more entries that contain item data describing the icon of an application. These entries include the coordinates that the system uses to display the icon; the count of bytes in the header, AND mask, and XOR mask for the icon; and the offset to the header, AND mask, and XOR mask for the icon.

The item data entries are followed by entries that contain the color data for the application icons. For more information about these entries, see Graphics Device Interface Overview.

For Windows version 3.1, the icon data is followed by tag data. The tag data contains information that Program Manager uses when it displays the Program Item Properties dialog box. This data identifies the directory in which the application is stored and the shortcut key (if one exists). It also specifies the state of the Run Minimized box.

### Group-File Structures

This topic uses C structures to depict the organization of data within a group file. These structures were created solely to show the organization of data in a resource; they do not appear in any of the include files shipped with the Microsoft Windows 3.1 Software Development Kit (SDK).

### Group-File Header

The group-file header contains general information about the group file. The GROUPHEADER structure has the following form:

```
struct tagGROUPHEADER {
    char    cIdentifier[4];
    WORD    wChecksum;
    WORD    cbGroup;
    WORD    nCmdShow;
    RECT    rcNormal;
    POINT   ptMin;
    WORD    pName;
    WORD    wLogPixelsX;
    WORD    wLogPixelsY;
    WORD    wBitsPerPixel;
    WORD    wPlanes;
    WORD    cItems;
    WORD    rgiItems[cItems];
};
```

Following are the members in the GROUPHEADER structure:

cIdentifier	Identifies an array of 4 characters. If the file is a valid group file, this array must contain the string "PMCC".
wChecksum	Specifies the negative sum of all words in the file (including the value specified by the wChecksum member).
cbGroup	Specifies the size of the group file, in bytes.
nCmdShow	Specifies whether Program Manager should display the group in minimized, normal, or maximized form. This member can be one of the following values:
Value	Flag
0x00	SW_HIDE
0x01	SW_SHOWNORMAL
0x02	SW_SHOWMINIMIZED
0x03	SW_SHOWMAXIMIZED
0x04	SW_SHOWNOACTIVATE

	0x05	SW_SHOW
	0x06	SW_MINIMIZE
	0x07	SW_SHOWMINNOACTIVATE
	0x08	SW_SHOWNA
	0x09	SW_RESTORE
rcNormal		Specifies the coordinates of the group window (the window in which the group icons appear). It is a rectangular structure.
ptMin		Specifies the coordinate of the lower-left corner of the group window with respect to the parent window. It is a point structure.
pName		Specifies an offset from the beginning of the file to a null-terminated string that specifies the group name.
wLogPixelsX		Specifies the horizontal resolution of the display for which the group icons were created.
wLogPixelsY		Specifies the vertical resolution of the display for which the group icons were created.
wBitsPerPixel		Specifies the format of the icon bitmaps, in bits per pixel.
wPlanes		Specifies the count of planes in the icon bitmaps.
cItems		Specifies the number of ITEMDATA structures in the rgItems array. This is not necessarily the number of items in the group, because there may be NULL entries in the rgItems array.
rgItems[cItems]		Specifies an array of ITEMDATA structures.

#### Item Data

The item data contains information about a particular application and its icon. The ITEMDATA structure has the following form:

```

struct tagITEMDATA {
    POINT pt;
    WORD iIcon;
    WORD cbResource;
    WORD cbANDPlane;
    WORD cbXORPlane;
    WORD pHeader;
    WORD pANDPlane;
    WORD pXORPlane;
    WORD pName;
    WORD pCommand;
    WORD pIconPath;
};

```

Following are the members in the ITEMDATA structure:

pt	Specifies the coordinates for the lower-left corner of an icon in the group window. It is a point structure.
iIcon	Specifies the index value for an icon. This value indicates the position of the icon in an executable file.
cbResource	Specifies the count of bytes in the icon resource, which appears in the executable file for the application.
cbANDPlane	Specifies the count of bytes in the AND mask for the icon.
cbXORPlane	Specifies the count of bytes in the XOR mask for the icon.
pHeader	Specifies an offset from the beginning of the group file to the resource header for the icon.
pANDPlane	Specifies an offset from the beginning of the group file to the AND mask for the icon.
pXORPlane	Specifies an offset from the beginning of the group file to the XOR mask for the icon.
pName	Specifies an offset from the beginning of the group file to a string that specifies the item name.
pCommand	Specifies an offset from the beginning of the group file to a string that specifies the name of the executable file containing the application and the icon resource(s).



pIconPath Specifies an offset from the beginning of the group file to a string that specifies the path where the executable file is located. This path can be used to extract icon data from an executable file.

#### Tag Data

The tag data contains general information used to display the Program Item Properties dialog box. The TAGDATA structure has the following form:

```
struct tagTAGDATA{  
    WORD wID;  
    WORD wItem;  
    WORD cb;  
    BYTE rgb[1];  
};
```

Following are the members in the TAGDATA structure:

wID	Specifies the type of tag data. This member can have one of the following values:								
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0x8101</td><td>Array at which the rgb member points is a null-terminated string that identifies the path for the application.</td></tr><tr><td>0x8102</td><td>Array at which the rgb member points is a 16-bit word value that identifies the shortcut key specified by the user.</td></tr><tr><td>0x8103</td><td>Minimized version of the item is displayed. If this value is specified, the array to which the rgb member points is not present in the structure and the value of the cb member is 0x06.</td></tr></tbody></table>	Value	Meaning	0x8101	Array at which the rgb member points is a null-terminated string that identifies the path for the application.	0x8102	Array at which the rgb member points is a 16-bit word value that identifies the shortcut key specified by the user.	0x8103	Minimized version of the item is displayed. If this value is specified, the array to which the rgb member points is not present in the structure and the value of the cb member is 0x06.
Value	Meaning								
0x8101	Array at which the rgb member points is a null-terminated string that identifies the path for the application.								
0x8102	Array at which the rgb member points is a 16-bit word value that identifies the shortcut key specified by the user.								
0x8103	Minimized version of the item is displayed. If this value is specified, the array to which the rgb member points is not present in the structure and the value of the cb member is 0x06.								
wItem	Specifies the index to the item the tag data refers to. If the data is not specific to a particular item, this value is 0xFFFF.								
cb	Specifies the size of the TAGDATA structure, in bytes.								
rgb	Specifies an array of byte values. The length of this array can be found by subtracting 6 from the value of the cb member.								

## Executable-File Header Format (3.1)

An executable (.EXE) file for the Microsoft Windows operating system contains a combination of code and data or a combination of code, data, and resources. The executable file also contains two headers: an MS-DOS header and a Windows header. The next two sections describe these headers; the third section describes the code and data contained in a Windows executable file.

### MS-DOS Header

The MS-DOS (old-style) executable-file header contains four distinct parts: a collection of header information (such as the signature word, the file size, and so on), a reserved section, a pointer to a Windows header (if one exists), and a stub program. The following illustration shows the MS-DOS executable-file header:

If the word value at offset 18h is 40h or greater, the word value at 3Ch is typically an offset to a Windows header. Applications must verify this for each executable-file header being tested, because a few applications have a different header style.

MS-DOS uses the stub program to display a message if Windows has not been loaded when the user attempts to run a program.

For more information about the MS-DOS executable-file header, see the Microsoft MS-DOS Programmer's Reference (Redmond, Washington: Microsoft Press, 1991).

### Windows Header

The Windows (new-style) executable-file header contains information that the loader requires for segmented executable files. This information includes the linker version number, data specified by the linker, data specified by the resource compiler, tables of segment data, tables of resource data, and so on. The following illustration shows the Windows executable-file header:

The following sections describe the entries in the Windows executable-file header.

#### Information Block

The information block in the Windows header contains the linker version number, the lengths of various tables that further describe the executable file, the offsets from the beginning of the header to the beginning of these tables, the heap and stack sizes, and so on. The following list summarizes the contents of the header information block (the locations are relative to the beginning of the block):

Location	Description																
00h	Specifies the signature word. The low byte contains "N" (4Eh) and the high byte contains "E" (45h).																
02h	Specifies the linker version number.																
03h	Specifies the linker revision number.																
04h	Specifies the offset to the entry table (relative to the beginning of the header).																
06h	Specifies the length of the entry table, in bytes.																
08h	Reserved.																
0Ch	Specifies flags that describe the contents of the executable file. This value can be one or more of the following bits: <table><thead><tr><th>Bit</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>The linker sets this bit if the executable-file format is SINGLEDATA. An executable file with this format contains one data segment. This bit is set if the file is a dynamic-link library (DLL).</td></tr><tr><td>1</td><td>The linker sets this bit if the executable-file format is MULTIPLEDATA. An executable file with this format contains multiple data segments. This bit is set if the file is a Windows application. If neither bit 0 nor bit 1 is set, the executable-file format is NOAUTODATA. An executable file with this format does not contain an automatic data segment.</td></tr><tr><td>2</td><td>Reserved.</td></tr><tr><td>3</td><td>Reserved.</td></tr><tr><td>8</td><td>Reserved.</td></tr><tr><td>9</td><td>Reserved.</td></tr><tr><td>11</td><td>If this bit is set, the first segment in the executable file contains code that loads the application.</td></tr></tbody></table>	Bit	Meaning	0	The linker sets this bit if the executable-file format is SINGLEDATA. An executable file with this format contains one data segment. This bit is set if the file is a dynamic-link library (DLL).	1	The linker sets this bit if the executable-file format is MULTIPLEDATA. An executable file with this format contains multiple data segments. This bit is set if the file is a Windows application. If neither bit 0 nor bit 1 is set, the executable-file format is NOAUTODATA. An executable file with this format does not contain an automatic data segment.	2	Reserved.	3	Reserved.	8	Reserved.	9	Reserved.	11	If this bit is set, the first segment in the executable file contains code that loads the application.
Bit	Meaning																
0	The linker sets this bit if the executable-file format is SINGLEDATA. An executable file with this format contains one data segment. This bit is set if the file is a dynamic-link library (DLL).																
1	The linker sets this bit if the executable-file format is MULTIPLEDATA. An executable file with this format contains multiple data segments. This bit is set if the file is a Windows application. If neither bit 0 nor bit 1 is set, the executable-file format is NOAUTODATA. An executable file with this format does not contain an automatic data segment.																
2	Reserved.																
3	Reserved.																
8	Reserved.																
9	Reserved.																
11	If this bit is set, the first segment in the executable file contains code that loads the application.																

13	If this bit is set, the linker detects errors at link time but still creates an executable file.
14	Reserved.
15	If this bit is set, the executable file is a library module. If bit 15 is set, the CS:IP registers point to an initialization procedure called with the value in the AX register equal to the module handle. The initialization procedure must execute a far return to the caller. If the procedure is successful, the value in AX is nonzero. Otherwise, the value in AX is zero. The value in the DS register is set to the library's data segment if SINGLEDATA is set. Otherwise, DS is set to the data segment of the application that loads the library.
0Eh	Specifies the automatic data segment number. (0Eh is zero if the SINGLEDATA and MULTIPLEDATA bits are cleared.)
10h	Specifies the initial size, in bytes, of the local heap. This value is zero if there is no local allocation.
12h	Specifies the initial size, in bytes, of the stack. This value is zero if the SS register value does not equal the DS register value.
14h	Specifies the segment:offset value of CS:IP.
18h	Specifies the segment:offset value of SS:SP. The value specified in SS is an index to the module's segment table. The first entry in the segment table corresponds to segment number 1. If SS addresses the automatic data segment and SP is zero, SP is set to the address obtained by adding the size of the automatic data segment to the size of the stack.
1Ch	Specifies the number of entries in the segment table.
1Eh	Specifies the number of entries in the module-reference table.
20h	Specifies the number of bytes in the nonresident-name table.
22h	Specifies a relative offset from the beginning of the Windows header to the beginning of the segment table.
24h	Specifies a relative offset from the beginning of the Windows header to the beginning of the resource table.
26h	Specifies a relative offset from the beginning of the Windows header to the beginning of the resident-name table.
28h	Specifies a relative offset from the beginning of the Windows header to the beginning of the module-reference table.
2Ah	Specifies a relative offset from the beginning of the Windows header to the beginning of the imported-name table.
2Ch	Specifies a relative offset from the beginning of the file to the beginning of the nonresident-name table.
30h	Specifies the number of movable entry points.
32h	Specifies a shift count that is used to align the logical sector. This count is log <sub>2</sub> of the segment sector size. It is typically 4, although the default count is 9. (This value corresponds to the /alignment [/a] linker switch. When the linker command line contains /a:16, the shift count is 4. When the linker command line contains /a:512, the shift count is 9.)
34h	Specifies the number of resource segments.
36h	Specifies the target operating system, depending on which bits are set: Bit    Meaning 0      Operating system format is unknown. 1      Reserved. 2      Operating system is Microsoft Windows. 3      Reserved. 4      Reserved.
37h	Specifies additional information about the executable file. It can be one or more of the following values: Bit    Meaning 1      If this bit is set, the executable file contains a Windows 2.x application that runs in version 3.x protected mode.

	2	If this bit is set, the executable file contains a Windows 2.x application that supports proportional fonts.
	3	If this bit is set, the executable file contains a fast-load area.
38h		Specifies the offset, in sectors, to the beginning of the fast-load area. (Only Windows uses this value.)
3Ah		Specifies the length, in sectors, of the fast-load area. (Only Windows uses this value.)
3Ch		Reserved.
3Eh		Specifies the expected version number for Windows. (Only Windows uses this value.)

#### Segment Table

The segment table contains information that describes each segment in an executable file. This information includes the segment length, segment type, and segment-relocation data. The following list summarizes the values found in the segment table (the locations are relative to the beginning of each entry):

Location	Description																																		
00h	Specifies the offset, in sectors, to the segment data (relative to the beginning of the file). A value of zero means no data exists.																																		
02h	Specifies the length, in bytes, of the segment, in the file. A value of zero indicates that the segment length is 64K, unless the selector offset is also zero.																																		
04h	Specifies flags that describe the contents of the executable file. This value can be one or more of the following: <table> <thead> <tr> <th>Bit</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>If this bit is set, the segment is a data segment. Otherwise, the segment is a code segment.</td> </tr> <tr> <td>1</td> <td>If this bit is set, the loader has allocated memory for the segment.</td> </tr> <tr> <td>2</td> <td>If this bit is set, the segment is loaded.</td> </tr> <tr> <td>3</td> <td>Reserved.</td> </tr> <tr> <td>4</td> <td>If this bit is set, the segment type is MOVABLE. Otherwise, the segment type is <u>FIXED</u>.</td> </tr> <tr> <td>5</td> <td>If this bit is set, the segment type is PURE or SHAREABLE. Otherwise, the segment type is IMPURE or NONSHAREABLE.</td> </tr> <tr> <td>6</td> <td>If this bit is set, the segment type is PRELOAD. Otherwise, the segment type is LOADONCALL.</td> </tr> <tr> <td>7</td> <td>If this bit is set and the segment is a code segment, the segment type is EXECUTEONLY. If this bit is set and the segment is a data segment, the segment type is READONLY.</td> </tr> <tr> <td>8</td> <td>If this bit is set, the segment contains relocation data.</td> </tr> <tr> <td>9</td> <td>Reserved.</td> </tr> <tr> <td>10</td> <td>Reserved.</td> </tr> <tr> <td>11</td> <td>Reserved.</td> </tr> <tr> <td>12</td> <td>If this bit is set, the segment is discardable.</td> </tr> <tr> <td>13</td> <td>Reserved.</td> </tr> <tr> <td>14</td> <td>Reserved.</td> </tr> <tr> <td>15</td> <td>Reserved.</td> </tr> </tbody> </table>	Bit	Meaning	0	If this bit is set, the segment is a data segment. Otherwise, the segment is a code segment.	1	If this bit is set, the loader has allocated memory for the segment.	2	If this bit is set, the segment is loaded.	3	Reserved.	4	If this bit is set, the segment type is MOVABLE. Otherwise, the segment type is <u>FIXED</u> .	5	If this bit is set, the segment type is PURE or SHAREABLE. Otherwise, the segment type is IMPURE or NONSHAREABLE.	6	If this bit is set, the segment type is PRELOAD. Otherwise, the segment type is LOADONCALL.	7	If this bit is set and the segment is a code segment, the segment type is EXECUTEONLY. If this bit is set and the segment is a data segment, the segment type is READONLY.	8	If this bit is set, the segment contains relocation data.	9	Reserved.	10	Reserved.	11	Reserved.	12	If this bit is set, the segment is discardable.	13	Reserved.	14	Reserved.	15	Reserved.
Bit	Meaning																																		
0	If this bit is set, the segment is a data segment. Otherwise, the segment is a code segment.																																		
1	If this bit is set, the loader has allocated memory for the segment.																																		
2	If this bit is set, the segment is loaded.																																		
3	Reserved.																																		
4	If this bit is set, the segment type is MOVABLE. Otherwise, the segment type is <u>FIXED</u> .																																		
5	If this bit is set, the segment type is PURE or SHAREABLE. Otherwise, the segment type is IMPURE or NONSHAREABLE.																																		
6	If this bit is set, the segment type is PRELOAD. Otherwise, the segment type is LOADONCALL.																																		
7	If this bit is set and the segment is a code segment, the segment type is EXECUTEONLY. If this bit is set and the segment is a data segment, the segment type is READONLY.																																		
8	If this bit is set, the segment contains relocation data.																																		
9	Reserved.																																		
10	Reserved.																																		
11	Reserved.																																		
12	If this bit is set, the segment is discardable.																																		
13	Reserved.																																		
14	Reserved.																																		
15	Reserved.																																		
06h	Specifies the minimum allocation size of the segment, in bytes. A value of zero indicates that the minimum allocation size is 64K.																																		

#### Resource Table

The resource table describes and identifies the location of each resource in the executable file. The table has the following form:

```
WORDrscAlignShift;
TYPEINFO rscTypes[];
WORDrscEndTypes;
BYTERscResourceNames[];
BYTERscEndNames;
```

Following are the members in the resource table:

rscAlignShift	Specifies the alignment shift count for resource data. When the shift count is used as an exponent of 2, the resulting value specifies the factor, in bytes, for computing the location of a resource in the executable file.
rscTypes	Specifies an array of TYPEINFO structures containing information about resource types. There must be one TYPEINFO structure for each type of resource in the executable file.
rscEndTypes	Specifies the end of the resource type definitions. This member must be zero.
rscResourceNames	Specifies the names (if any) associated with the resources in this table. Each name is stored as consecutive bytes; the first byte specifies the number of characters in the name.
rscEndNames	Specifies the end of the resource names and the end of the resource table. This member must be zero.

### Type Information

The TYPEINFO structure has the following form:

```
typedef struct _TYPEINFO {
    WORD    rtTypeID;
    WORD    rtResourceCount;
    DWORD   rtReserved;
    NAMEINFO rtNameInfo[ ];
} TYPEINFO;
```

Following are the members in the TYPEINFO structure:

rtTypeID	Specifies the type identifier of the resource. This integer value is either a resource-type value or an offset to a resource-type name. If the high bit in this member is set (0x8000), the value is one of the following resource-type values:																										
	<table> <thead> <tr> <th>Value</th> <th>Resource type</th> </tr> </thead> <tbody> <tr> <td>RT_ACCELERATOR</td> <td>Accelerator table</td> </tr> <tr> <td>RT_BITMAP</td> <td>Bitmap</td> </tr> <tr> <td>RT_CURSOR</td> <td>Cursor</td> </tr> <tr> <td>RT_DIALOG</td> <td>Dialog box</td> </tr> <tr> <td>RT_FONT</td> <td>Font component</td> </tr> <tr> <td>RT_FONTDIR</td> <td>Font directory</td> </tr> <tr> <td>RT_GROUP_CURSOR</td> <td>Cursor directory</td> </tr> <tr> <td>RT_GROUP_ICON</td> <td>Icon directory</td> </tr> <tr> <td>RT_ICON</td> <td>Icon</td> </tr> <tr> <td>RT_MENU</td> <td>Menu</td> </tr> <tr> <td>RT_RCDATA</td> <td>Resource data</td> </tr> <tr> <td>RT_STRING</td> <td>String table</td> </tr> </tbody> </table>	Value	Resource type	RT_ACCELERATOR	Accelerator table	RT_BITMAP	Bitmap	RT_CURSOR	Cursor	RT_DIALOG	Dialog box	RT_FONT	Font component	RT_FONTDIR	Font directory	RT_GROUP_CURSOR	Cursor directory	RT_GROUP_ICON	Icon directory	RT_ICON	Icon	RT_MENU	Menu	RT_RCDATA	Resource data	RT_STRING	String table
Value	Resource type																										
RT_ACCELERATOR	Accelerator table																										
RT_BITMAP	Bitmap																										
RT_CURSOR	Cursor																										
RT_DIALOG	Dialog box																										
RT_FONT	Font component																										
RT_FONTDIR	Font directory																										
RT_GROUP_CURSOR	Cursor directory																										
RT_GROUP_ICON	Icon directory																										
RT_ICON	Icon																										
RT_MENU	Menu																										
RT_RCDATA	Resource data																										
RT_STRING	String table																										
	If the high bit of the value in this member is not set, the value represents an offset, in bytes relative to the beginning of the resource table, to a name in the rscResourceNames member.																										
rtResourceCount	Specifies the number of resources of this type in the executable file.																										
rtReserved	Reserved.																										
rtNameInfo	Specifies an array of NAMEINFO structures containing information about individual resources. The rtResourceCount member specifies the number of structures in the array.																										

### Name Information

The NAMEINFO structure has the following form:

```
typedef struct _NAMEINFO {
    WORD rnOffset;
    WORD rnLength;
    WORD rnFlags;
    WORD rnID;
    WORD rnHandle;
    WORD rnUsage;
```

```
} NAMEINFO;
```

Following are the members in the NAMEINFO structure:

mOffset	Specifies an offset to the contents of the resource data (relative to the beginning of the file). The offset is in terms of alignment units specified by the rscAlignShift member at the beginning of the resource table.								
mLength	Specifies the resource length, in bytes.								
mFlags	Specifies whether the resource is fixed, preloaded, or shareable. This member can be one or more of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0x0010</td><td>Resource is movable (MOVEABLE). Otherwise, it is fixed.</td></tr><tr><td>0x0020</td><td>Resource can be shared (PURE).</td></tr><tr><td>0x0040</td><td>Resource is preloaded (PRELOAD). Otherwise, it is loaded on demand.</td></tr></tbody></table>	Value	Meaning	0x0010	Resource is movable (MOVEABLE). Otherwise, it is fixed.	0x0020	Resource can be shared (PURE).	0x0040	Resource is preloaded (PRELOAD). Otherwise, it is loaded on demand.
Value	Meaning								
0x0010	Resource is movable (MOVEABLE). Otherwise, it is fixed.								
0x0020	Resource can be shared (PURE).								
0x0040	Resource is preloaded (PRELOAD). Otherwise, it is loaded on demand.								
mID	Specifies or points to the resource identifier. If the identifier is an integer, the high bit is set (8000h). Otherwise, it is an offset to a resource string, relative to the beginning of the resource table.								
mHandle	Reserved.								
mUsage	Reserved.								

#### Resident-Name Table

The resident-name table contains strings that identify exported functions in the executable file. As the name implies, these strings are resident in system memory and are never discarded. The resident-name strings are case-sensitive and are not null-terminated. The following list summarizes the values found in the resident-name table (the locations are relative to the beginning of each entry):

Location	Description
00h	Specifies the length of a string. If there are no more strings in the table, this value is zero.
01h - xxh	Specifies the resident-name text. This string is case-sensitive and is not null-terminated.
xxh + 01h	Specifies an ordinal number that identifies the string. This number is an index into the entry table.

The first string in the resident-name table is the module name.

#### Module-Reference Table

The module-reference table contains offsets for module names stored in the imported-name table. Each entry in this table is 2 bytes long.

#### Imported-Name Table

The imported-name table contains the names of modules that the executable file imports. Each entry contains two parts: a single byte that specifies the length of the string and the string itself. The strings in this table are not null-terminated.

#### Entry Table

The entry table contains bundles of entry points from the executable file (the linker generates each bundle). The numbering system for these ordinal values is 1-based--that is, the ordinal value corresponding to the first entry point is 1.

The linker generates the densest possible bundles under the restriction that it cannot reorder the entry points. This restriction is necessary because other executable files may refer to entry points within a given bundle by their ordinal values.

The entry-table data is organized by bundle, each of which begins with a 2-byte header. The first byte of the header specifies the number of entries in the bundle (a value of 00h designates the end of the table). The second byte specifies whether the corresponding segment is movable or fixed. If the value in this byte is 0FFh, the segment is movable. If the value in this byte is 0FEh, the entry does not refer to a segment but refers, instead, to a constant defined within the module. If the value in this byte is neither 0FFh nor 0FEh, it is a segment index.

For movable segments, each entry consists of 6 bytes and has the following form:

Location	Description				
00h	Specifies a byte value. This value can be a combination of the following bits: <table><thead><tr><th>Bit(s)</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>If this bit is set, the entry is exported.</td></tr></tbody></table>	Bit(s)	Meaning	0	If this bit is set, the entry is exported.
Bit(s)	Meaning				
0	If this bit is set, the entry is exported.				

1	If this bit is set, the segment uses a global (shared) data segment.
3-7	If the executable file contains code that performs ring transitions, these bits specify the number of words that compose the stack. At the time of the ring transition, these words must be copied from one ring to the other.
01h	Specifies an int 3fh instruction.
03h	Specifies the segment number.
04h	Specifies the segment offset.

For fixed segments, each entry consists of 3 bytes and has the following form:

Location	Description								
00h	Specifies a byte value. This value can be a combination of the following bits:								
	<table> <thead> <tr> <th>Bit(s)</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>If this bit is set, the entry is exported.</td> </tr> <tr> <td>1</td> <td>If this bit is set, the entry uses a global (shared) data segment. (This may be set only for SINGLEDATA library modules.)</td> </tr> <tr> <td>3-7</td> <td>If the executable file contains code that performs ring transitions, these bits specify the number of words that compose the stack. At the time of the ring transition, these words must be copied from one ring to the other.</td> </tr> </tbody> </table>	Bit(s)	Meaning	0	If this bit is set, the entry is exported.	1	If this bit is set, the entry uses a global (shared) data segment. (This may be set only for SINGLEDATA library modules.)	3-7	If the executable file contains code that performs ring transitions, these bits specify the number of words that compose the stack. At the time of the ring transition, these words must be copied from one ring to the other.
Bit(s)	Meaning								
0	If this bit is set, the entry is exported.								
1	If this bit is set, the entry uses a global (shared) data segment. (This may be set only for SINGLEDATA library modules.)								
3-7	If the executable file contains code that performs ring transitions, these bits specify the number of words that compose the stack. At the time of the ring transition, these words must be copied from one ring to the other.								
01h	Specifies an offset.								

#### Nonresident-Name Table

The nonresident-name table contains strings that identify exported functions in the executable file. As the name implies, these strings are not always resident in system memory and are discardable. The nonresident-name strings are case-sensitive; they are not null-terminated. The following list summarizes the values found in the nonresident-name table (the specified locations are relative to the beginning of each entry):

Location	Description
00h	Specifies the length, in bytes, of a string. If this byte is 00h, there are no more strings in the table.
01h - xxh	Specifies the nonresident-name text. This string is case-sensitive and is not null-terminated.
xx + 01h	Specifies an ordinal number that is an index to the entry table.

The first name that appears in the nonresident-name table is the module description string (which was specified in the module-definition file).

#### Code Segments and Relocation Data

Code and data segments follow the Windows header. Some of the code segments may contain calls to functions in other segments and may, therefore, require relocation data to resolve those references. This relocation data is stored in a relocation table that appears immediately after the code or data in the segment. The first 2 bytes in this table specify the number of relocation items the table contains. A relocation item is a collection of bytes specifying the following information:

- Address type (segment only, offset only, segment and offset)
- Relocation type (internal reference, imported ordinal, imported name)
- Segment number or ordinal identifier (for internal references)
- Reference-table index or function ordinal number (for imported ordinals)
- Reference-table index or name-table offset (for imported names)

Each relocation item contains 8 bytes of data, the first byte of which specifies one of the following relocation-address types:

Value	Meaning
0	Low byte at the specified offset
2	16-bit selector
3	32-bit pointer
5	16-bit offset
11	48-bit pointer
13	32-bit offset

The second byte specifies one of the following relocation types:

Value	Meaning
-------	---------

- 0 Internal reference
- 1 Imported ordinal
- 2 Imported name
- 3 OSFIXUP

The third and fourth bytes specify the offset of the relocation item within the segment.

If the relocation type is imported ordinal, the fifth and sixth bytes specify an index to a module's reference table and the seventh and eighth bytes specify a function ordinal value.

If the relocation type is imported name, the fifth and sixth bytes specify an index to a module's reference table and the seventh and eighth bytes specify an offset to an imported-name table.

If the relocation type is internal reference and the segment is fixed, the fifth byte specifies the segment number, the sixth byte is zero, and the seventh and eighth bytes specify an offset to the segment. If the relocation type is internal reference and the segment is movable, the fifth byte specifies 0FFh, the sixth byte is zero; and the seventh and eighth bytes specify an ordinal value found in the segment's entry table.



## Resource Format Overview (3.1)

This topic describes the format of executable-file resources used by the Microsoft Windows operating system. A resource, or collection of binary data, can be one of two types: standard or user-defined. The data in a standard resource describes an icon, cursor, menu, dialog box, bitmap, font, string table, or accelerator. The data in a user-defined resource describes an application-specific object. This topic describes standard resources.

A Windows executable file contains a resource table that describes each of the resources in the file. The data in this table includes an offset from the beginning of the file to each resource. It also includes values that specify the resource type, the resource length, and so on. For more information about the organization of the resource table, see Executable-File Header Format.

This topic uses C structures to depict the organization of data in resources. In some cases, these structures are not true C structures, because they contain members that can be variable-length strings. These structures were created only to depict the organization of data within a resource; they do not appear in any of the include files shipped with the Microsoft Windows 3.1 Software Development Kit (SDK).

[Icon Resource](#)

[Cursor Resource](#)

[Menu Resource](#)

[Dialog Box Resource](#)

[Bitmap Resource](#)

[Font Resource](#)

[String-Table Resources](#)

[Accelerator Resource](#)

[Name-Table Resource](#)

[Version-Information Resource](#)

### Icon Resource (3.1)

An icon resource is identical in format to an icon image in an icon-resource file. The resource contains the icon-image header, color table, and XOR and AND masks. For more information about the icon-image format, see Graphics Device Interface Overview.

Each icon resource must have a corresponding entry in the resource table of the executable file. This means the resource table must contain a TYPEINFO structure in which the rscTypeID member is set to the RT\_ICON value.

#### Icon-Directory Resource

An icon-directory resource is nearly identical in format to an icon directory in an icon-resource file. The resource specifies the number of icon images associated with this resource, as well as the dimensions and color formats for each icon. However, the last member of the ICONDIRENTRY structure (dwImageOffset) is replaced with a 16-bit value that specifies the resource-table index of the corresponding icon-image resource. The index is 1-based. If an executable file contains multiple icon resources, the index must be unique across all directories. For more information about the icon-directory format, see Graphics Device Interface Overview.

Each icon-directory resource must have a corresponding entry in the resource table of the executable file. This means the resource table must contain a TYPEINFO structure in which the rscTypeID member is set to the RT\_GROUP\_ICON value.

### Cursor Resource (3.1)

A cursor resource is nearly identical in format to a cursor image in a cursor-resource file. The resource contains the cursor hot spot as well as the cursor-image header, color table, and XOR and AND masks. The x- and y-coordinates for the cursor hot spot (both 16-bit values) appear first in the resource, immediately followed by the cursor-image header. For more information about the cursor-image format, see Graphics Device Interface Overview.

Each cursor resource must have a corresponding entry in the resource table of the executable file. This means the resource table must contain a TYPEINFO structure in which the rscTypeID member is set to the RT\_CURSOR value.

#### Cursor-Directory Resource

A cursor-directory resource is nearly identical in format to a cursor directory in a cursor-resource file. The resource specifies the number of cursor images associated with this resource, as well as the dimensions of the images, but it does not include the hot-spot data. Furthermore, the last member of the ICONDIRENTRY structure (dwImageOffset) is replaced with a 16-bit value that specifies the resource-table index of the corresponding cursor-image resource.

In an executable file, the CURSORDIRENTRY structure has the following form:

```
typedef struct _CURSORDIRENTRY {
    WORD    wWidth;
    WORD    wHeight;
    WORD    wPlanes;
    WORD    wBitCount;
    DWORD   lBytesInRes;
    WORD    wImageIndex;
} CURSORDIRENTRY;
```

Following are the members in the CURSORDIRENTRY structure:

wWidth	Specifies the width of the cursor, in pixels.
wHeight	Specifies the height of the cursor, in pixels.
wPlanes	Specifies the number of color planes in the bitmap. This member must be set to 1.
wBitCount	Specifies the number of color bits per pixel in the bitmap. This member must be set to 1.
lBytesInRes	Specifies the size of the resource, in bytes.
wImageIndex	Specifies the 1-based index identifying the cursor image associated with this cursor-directory resource. If an executable file contains multiple icon resources, the index must be unique across all directories.

Each cursor-directory resource must have a corresponding entry in the resource table of the executable file. This means the resource table must contain a TYPEINFO structure in which the rscTypeID member is set to the RT\_GROUP\_CURSOR value.

## Menu Resource (3.1)

A menu resource contains a header followed by a list of normal and pop-up menu items.

Each entry in the executable file's resource table contains a member that identifies the resource type. The `RT_MENU` constant identifies a menu resource.

### Menu Header

The menu header contains version information for the menu resource. The header consists of two 16-bit values (which must be zero for Windows version 3.0 and later). A `MenuHeader` structure has the following form:

```
struct MenuHeader {
    WORD wVersion;
    WORD wReserved;
};
```

Following are the members in the `MenuHeader` structure:

`wVersion` Specifies the version number. (For Windows 3.0 and later, this value is zero.)  
`wReserved` Reserved; must be zero.

### Pop-up Menu Item

A menu resource contains data for each pop-up item in a menu. The first 16 bits indicate whether the item is grayed, inactive, checked, and so on. This data also includes a string that appears in the rectangle corresponding to that item. A `PopupMenuItem` structure has the following form:

```
struct PopupMenuItem {
    WORD fItemFlags;
    char szItemText[];
};
```

Following are the members in the `PopupMenuItem` structure:

`fItemFlags` Specifies menu-item information. This member can have one or more of the following values:

Value	Meaning
<code>MF_GRAYED</code>	Item is grayed.
<code>MF_DISABLED</code>	Item is inactive.
<code>MF_CHECKED</code>	Item can be checked.
<code>MF_POPUP</code>	Item is a popup (must be specified for pop-up items).
<code>MF_MENUBARBREAK</code>	Item is a menu-bar break.
<code>MF_MENUBREAK</code>	Item is a menu break.
<code>MF_END</code>	Item ends the menu.

`szItemText` Specifies a null-terminated string that appears in the menu and identifies the menu item. There is no fixed limit on the size of this string.

### Normal Menu Item

A normal menu item is very similar to a pop-up menu item, except that it has an additional menu identifier. A `NormalMenuItem` structure has the following form:

```
struct NormalMenuItem {
    WORD fItemFlags;
    WORD wMenuID;
    char szItemText[];
};
```

Following are the members in the `NormalMenuItem` structure:

`fItemFlags` Specifies menu-item information. This member can have one or more of the following values:

Value	Meaning
<code>MF_GRAYED</code>	Item is grayed.
<code>MF_DISABLED</code>	Item is inactive.
<code>MF_CHECKED</code>	Item can be checked.

	MF_MENUBARBREAK	Item is a menu-bar break.
	MF_MENUBREAK	Item is a menu break.
	MF_END	Item ends the menu.
wMenuID	Identifies the menu item.	
szItemText	Specifies a null-terminated string that appears in the menu and identifies the menu item. There is no fixed limit on the size of this string.	

A menu separator is a normal menu item for which fItemFlags is zero, wMenuID is zero, and the szItemText array is empty.

#### Combined Menu Items

Pop-up and normal menu items are often combined in menus. A mixture of the two is shown in the following example:

```

POPUP ITEM
  NORMAL ITEM
  NORMAL ITEM
  .
  .
  .
  NORMAL ITEM
  NORMAL ITEM (fItemFlags contains the MF_END constant)

```

Note that the terminating item is a normal menu item, not a pop-up item, and that the fItemFlags member in the last item contains the MF\_END constant.

Pop-up and normal menu items can also be nested to create hierarchical blocks, as shown in the following example:

```

POPUP ITEM
  NORMAL ITEM
  NORMAL ITEM
  .
  .
  .
  NORMAL ITEM
  POPUP ITEM
  NORMAL ITEM
  NORMAL ITEM
  NORMAL ITEM
  POPUP ITEM (fItemFlags contains the MF_END constant)
  NORMAL ITEM
  NORMAL ITEM (fItemFlags contains the MF_END constant)
  NORMAL ITEM (fItemFlags contains the MF_END constant)

```

Note that, although the pop-up menu item has its own terminating item, the terminating item for the entire menu is again a normal menu item.

## Dialog Box Resource (3.1)

A dialog box resource contains a dialog box header and data for each control within the dialog box.

Each entry in the executable file's resource table contains a member that identifies the resource type. The `RT_DIALOG` constant identifies a dialog box resource.

### Dialog Box Header

The dialog box header contains general dialog box data, such as the dialog box window style, the number of controls in the dialog box, the coordinates of the upper-left corner of the box, the width and height of the box, the name of the menu to be displayed, and so on. The `DialogBoxHeader` structure has the following form:

```
struct DialogBoxHeader {
    DWORD lStyle;
    BYTE  bNumberOfItems;
    WORD  x;
    WORD  y;
    WORD  cx;
    WORD  cy;
    char  szMenuName[];
    char  szClassName[];
    char  szCaption[];
    WORD  wPointSize;    /* only if DS_SETFONT */
    char  szFaceName[]; /* only if DS_SETFONT */
};
```

Following are the members in the `DialogBoxHeader` structure:

<code>lStyle</code>	Specifies the dialog-window style. This member is a combination of the window-style and dialog-style flags that are found in the <code>WINDOWS.H</code> include file.
<code>bNumberOfItems</code>	Specifies the number of controls in the dialog box.
<code>x</code>	Specifies the x-coordinate of the upper-left corner of the dialog box. This coordinate is a horizontal distance from the left edge of the parent window. This distance is specified by using a special horizontal dialog box unit equivalent to the average character width of the font divided by 4. If the <code>DS_SETFONT</code> flag is set, the average character width of the font specified in the dialog box header is used. Otherwise, the average character width of the system font is used.
<code>y</code>	Specifies the y-coordinate of the lower-left corner of the dialog box. This coordinate is a vertical distance from the top of the parent window. This distance is specified by using a special vertical dialog box unit equivalent to the character height of the current font divided by 8. If the <code>DS_SETFONT</code> flag is set, the height of the font specified in the dialog box header is used. Otherwise, the height of the system font is used.
<code>cx</code>	Specifies the width of the dialog box, in horizontal dialog units. (See the description of the <code>x</code> member for a definition of horizontal dialog units.)
<code>cy</code>	Specifies the height of the dialog box, in vertical dialog units. (See the description of the <code>y</code> member for a definition of vertical dialog units.)
<code>szMenuName</code>	Identifies a menu resource associated with the dialog box. If no menu is associated with the dialog box, this array contains a single-byte value of zero. If the menu has an ordinal identifier, the first byte of this member contains <code>0xFF</code> and the subsequent two bytes contain the ordinal value. If the menu has a name identifier, the member contains a null-terminated string that specifies the menu name.
<code>szClassName</code>	Specifies the class name for the dialog box. If the dialog box uses the default class, this member contains a single-byte value of zero. Otherwise, this member contains a null-terminated string that specifies the name of the dialog class.
<code>szCaption</code>	Specifies a dialog box caption. This array must contain a null-terminated string.
<code>wPointSize</code>	Specifies the point size of a font that is unique to the dialog box. (This member is present only if the <code>DS_SETFONT</code> flag is set by the <code>lStyle</code> member.)

szFaceName Specifies the typeface name of a dialog box font. This array must contain a null-terminated string. (This member is present only if the DS\_SETFONT flag is set by the lStyle member.)

#### Control Data

A dialog box resource contains data for each control in a given dialog box. This data contains the coordinates of the upper-left corner of the control, the dimensions of the control, a control identifier, and so on. A ControlData structure has the following form:

```
struct ControlData {
    WORD x;
    WORD y;
    WORD cx;
    WORD cy;
    WORD wID;
    DWORD lStyle;
    union
    {
        BYTE class; /* if (class & 0x80) */
        char szClass[]; /* otherwise */
    } ClassID;
    szText;
};
```

Following are the members in the ControlData structure:

x	Specifies the x-coordinate of the upper-left corner of the control.														
y	Specifies the y-coordinate of the upper-left corner of the control.														
cx	Specifies the width of the control, in horizontal dialog box units. For a definition of these units, see the DialogBoxHeader structure in the preceding section.														
cy	Specifies the height of the control, in vertical dialog box units. For a definition of these units, see the DialogBoxHeader structure in the preceding section.														
wID	Identifies the control.														
lStyle	Specifies the control style. This member is a combination of the window-style flags that appear in the WINDOWS.H file.														
ClassID	Specifies the class type. This member is either a single-byte value or a null-terminated string. If this member is a byte value, it can be one of the following: <table><tr><th>Value</th><th>Class type</th></tr><tr><td>0x80</td><td>Button</td></tr><tr><td>0x81</td><td>Edit</td></tr><tr><td>0x82</td><td>Static</td></tr><tr><td>0x83</td><td>List box</td></tr><tr><td>0x84</td><td>Scroll bar</td></tr><tr><td>0x85</td><td>Combo box</td></tr></table>	Value	Class type	0x80	Button	0x81	Edit	0x82	Static	0x83	List box	0x84	Scroll bar	0x85	Combo box
Value	Class type														
0x80	Button														
0x81	Edit														
0x82	Static														
0x83	List box														
0x84	Scroll bar														
0x85	Combo box														
szClass	Identifies the class type. This member is a null-terminated string.														
szText	Specifies the control text. This member is a null-terminated string.														

### Bitmap Resource (3.1)

A bitmap resource is identical in format to a Windows bitmap file with its BITMAPFILEHEADER structure removed. In other words, the bitmap resource contains only the bitmap header, color table, and bitmap bits. For more information about the bitmap format, see Graphics Device Interface Overview.

Each bitmap resource must have a corresponding entry in the resource table of the executable file. This means the resource table must contain a TYPEINFO structure in which the rscTypeID member is set to the RT\_BITMAP value.



## Font Resource (3.1)

A font resource consists of two parts: a directory and its components. The font-directory data describes all the fonts in a resource. This data includes a value specifying the number of fonts in the resource and a table of metrics for each of these fonts. The font-component data describes a single font in the resource. There is one component for each of the fonts in the resource. The component data is identical to the data found in a Windows font file (.FNT).

Each entry in the executable file's resource table contains a member that identifies the resource type. The `RT_FONTDIR` and `RT_FONT` constants identify a font directory and a font component, respectively.

### Font-Directory Data

Font-directory data consists of a font count and one or more font directory entries.

### Font Count

The font count is an integer that specifies the number of fonts in the resource. This value also corresponds to the number of font directories and font components.

### Font Directory

The font directory is a collection of font metrics for a particular font. These metrics specify the point size for the font, aspect ratio, stroke width, and so on. The `FontDirEntry` structure has the following form:

```
struct FontDirEntry {
    WORD    fontOrdinal;
    WORD    dfVersion;
    DWORD   dfSize;
    char    dfCopyright[60];
    WORD    dfType;
    WORD    dfPoints;
    WORD    dfVertRes;
    WORD    dfHorizRes;
    WORD    dfAscent;
    WORD    dfInternalLeading;
    WORD    dfExternalLeading;
    BYTE    dfItalic;
    BYTE    dfUnderline;
    BYTE    dfStrikeOut;
    WORD    dfWeight;
    BYTE    dfCharSet;
    WORD    dfPixWidth;
    WORD    dfPixHeight;
    BYTE    dfPitchAndFamily;
    WORD    dfAvgWidth;
    WORD    dfMaxWidth;
    BYTE    dfFirstChar;
    BYTE    dfLastChar;
    BYTE    dfDefaultChar;
    BYTE    dfBreakChar;
    WORD    dfWidthBytes;
    DWORD   dfDevice;
    DWORD   dfFace;
    DWORD   dfReserved;
    char    szDeviceName[];
    char    szFaceName[];
};
```

### Font-Component Data

Font-component data consists of one or more font-component entries.

### Font Component

Each font-component entry consists of a header, extension data, extended text metrics, kerning-pair data, and track-kerning data.

Following are the five parts of the font component entries:

Data structure	Contents
Header	Font metrics, such as the aspect ratio for which the font was created; leading values; italic, underline, strikeout, and bold descriptions; width information; first and last character identifiers; default and break character identifiers; and a pointer to the actual character data
Extension data	Offset to the extended font metrics, offset to the extent table, offset to the origin table, and offset to the table of kerning data
Extended text metrics	Additional font metrics, such as the point size of the font, the minimum point size to which it can be scaled, the maximum point size to which it can be scaled, the "X" height, the lowercase ascent and descent values, superscript metrics and offsets, subscript metrics and offsets, underline offset and width, strikeout offset and width, and the number of kerning pairs associated with the font
Kerning-pair data	An identifier for each character in the pair of kerned characters, and a kerning value
Track-kerning data	Additional kerning data

For a complete description of Windows font files, see the Microsoft Windows Device Development Kit documentation.

### String-Table Resources (3.1)

A string table consists of one or more separate resources, each containing exactly 16 strings. The maximum length of each string is 255 bytes. One or more strings in a block can be null or empty. The first byte in the string specifies the number of characters in the string. (For null or empty strings, the first byte contains the value zero.)

Windows uses a 16-bit identifier to locate a string in a string-table resource. Bits 4 through 15 specify the block in which the string appears; bits 0 through 3 specify the location of that string relative to the beginning of the block.

Each entry in an executable file's resource table contains a member that identifies the resource type. The RT\_STRING constant identifies a string table.

### Accelerator Resource (3.1)

An accelerator resource contains one or more accelerator entries.

Each entry in an executable file's resource table contains a member that identifies the resource type. The `RT_ACCELERATOR` constant identifies an accelerator resource.

The accelerator entry is a 5-byte entry with the following form:

```
struct AccelTableEntry {
    BYTE fFlags;
    WORD wEvent;
    WORD wID;
};
```

Following are the members in the `AccelTableEntry` structure:

<code>fFlags</code>	Specifies accelerator characteristics. It can be one or more of the following values:												
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0x02</td><td>Top-level menu item is not highlighted when accelerator is used.</td></tr><tr><td>0x04</td><td>Accelerator is activated only if user presses the SHIFT key. This flag applies only to virtual keys.</td></tr><tr><td>0x08</td><td>Accelerator is activated only if user presses the <u>CONTROL</u> key. This flag applies only to virtual keys.</td></tr><tr><td>0x10</td><td>Accelerator is activated only if user presses the ALT key. This flag applies only to virtual keys.</td></tr><tr><td>0x80</td><td>Entry is last entry in accelerator table.</td></tr></tbody></table>	Value	Meaning	0x02	Top-level menu item is not highlighted when accelerator is used.	0x04	Accelerator is activated only if user presses the SHIFT key. This flag applies only to virtual keys.	0x08	Accelerator is activated only if user presses the <u>CONTROL</u> key. This flag applies only to virtual keys.	0x10	Accelerator is activated only if user presses the ALT key. This flag applies only to virtual keys.	0x80	Entry is last entry in accelerator table.
Value	Meaning												
0x02	Top-level menu item is not highlighted when accelerator is used.												
0x04	Accelerator is activated only if user presses the SHIFT key. This flag applies only to virtual keys.												
0x08	Accelerator is activated only if user presses the <u>CONTROL</u> key. This flag applies only to virtual keys.												
0x10	Accelerator is activated only if user presses the ALT key. This flag applies only to virtual keys.												
0x80	Entry is last entry in accelerator table.												
<code>wEvent</code>	Specifies an ASCII character value or a virtual-key code that identifies the accelerator key.												
<code>wID</code>	Identifies the accelerator. This is the value passed to the window procedure when the user presses the key.												

### Name-Table Resource (3.1)

Name-table entries are not used in Windows 3.1. They are supported in Windows 3.0, but they can adversely affect system performance.

The header in a Windows executable file contains a resource table. This table contains data that describes many of the resources in the file. In Windows 3.0, the resource table does not describe named resources or resources that use a type name as a unique identifier. Instead, a name-table structure in the resource table maps a unique integer value to each resource name or type.

Each entry in an executable file's resource table contains a member that identifies the resource type. The decimal value 15 identifies a name-table resource.

## Version-Information Resource (3.1)

A version-information resource contains data that identifies the version, language, and distribution of the application, dynamic-link library, driver, or device containing the resource. Installation programs use the functions in the File Installation library (VER.DLL) to retrieve the version-information resource from a file and to extract the version-information blocks from the resource. (For more information about the File Installation library, see the Microsoft Windows Programmer's Reference, Volume 1.)

A version-information resource consists of one or more information blocks, each with the following form:

```
WORD    cbBlock;  
WORD    cbValue;  
char    szKey[ ];  
BYTE    abValue[ ];
```

Following are the members in a version-information block:

cbBlock	Specifies the size, in bytes, of the complete block. This value includes the size of nested blocks, if any.
cbValue	Specifies the size, in bytes, of the abValue member.
szKey	Specifies the name of the block. This value is a null-terminated string. Additional zero bytes are appended to the string to align the last byte on a 32-bit boundary.
abValue	Specifies either an array of word values or a null-terminated string. The format of this member depends on the szKey value. Additional zero bytes are appended to align the last byte on a 32-bit boundary.

A block can contain nested blocks. In such cases, the nested block immediately follows the abValue member and the size specified by the cbBlock member in the first block is the sum of the two sizes. If a block contains more than one nested block, the nested blocks are stored sequentially and the cbBlock member in the first block specifies the total size of all blocks.

A version-information resource usually contains the following predefined blocks:

- Root
- Variable information
- String information
- Language-specific

In addition, the string and variable information blocks usually contain nested blocks that define the details about the file. This section describes the predefined information blocks.

### Root Block

A root block is always the first block in the version resource. It contains such information as the file version, product version, release status, operating system, file type, and date the file was created.

The name of the root block, as specified by the szKey member, is `VS_VERSION_INFO`. The value (in abValue) is a `VS_FIXEDFILEINFO` structure. For a description of the `VS_FIXEDFILEINFO` structure, see the Microsoft Windows Programmer's Reference, Volume 3.

The variable and string information blocks in the resource are nested within the root block.

### Variable Information Block

A variable information block typically contains a single nested block that defines the languages and character sets supported by the file.

The variable information block has the name `VarFileInfo` but has no corresponding value. Instead, the block is immediately followed by a nested block that has the name `Translation` and has a value consisting of an array of language and character-set identifiers. Each element in the array consists of two 16-bit values. The first value is a language identifier, the second a character-set identifier.

The language identifier can be one of the following values:

Value	Language
0x0401	Arabic
0x0402	Bulgarian
0x0403	Catalan
0x0404	Traditional Chinese
0x0405	Czech
0x0406	Danish

0x0407	German
0x0408	Greek
0x0409	U.S. English
0x040A	Castilian Spanish
0x040B	Finnish
0x040C	French
0x040D	Hebrew
0x040E	Hungarian
0x040F	Icelandic
0x0410	Italian
0x0411	Japanese
0x0412	Korean
0x0413	Dutch
0x0414	Norwegian - Bokmål
0x0415	Polish
0x0416	Brazilian Portuguese
0x0417	Rhaeto-Romanic
0x0418	Romanian
0x0419	Russian
0x041A	Croato-Serbian (Latin)
0x041B	Slovak
0x041C	Albanian
0x041D	Swedish
0x041E	Thai
0x041F	Turkish
0x0420	Urdu
0x0421	Bahasa
0x0804	Simplified Chinese
0x0807	Swiss German
0x0809	U.K. English
0x080A	Mexican Spanish
0x080C	Belgian French
0x0810	Swiss Italian
0x0813	Belgian Dutch
0x0814	Norwegian - Nynorsk
0x0816	Portuguese
0x081A	Serbo-Croatian (Cyrillic)
0x0C0C	Canadian French
0x100C	Swiss French

The character-set identifier can be one of the following values:

Value	Character set
0	7-bit ASCII
932	Windows, Japan (Shift - JIS X-0208)
949	Windows, Korea (Shift - KSC 5601)
950	Windows, Taiwan (GB5)
1200	Unicode
1250	Windows, Latin-2 (Eastern European)
1251	Windows, Cyrillic
1252	Windows, Multilingual
1253	Windows, Greek
1254	Windows, Turkish
1255	Windows, Hebrew

Character set 1252 is typically given for files designed for the U.S. English version of Windows.

### String Information Block

A string information block contains version information in the form of null-terminated strings.

The string information block has the name `StringFileInfo` but has no corresponding value. Instead, the block contains one or more nested blocks. Each nested block corresponds to one pair of language and character-set identifiers given in the variable information block.

### Language-Specific Blocks

A language-specific block contains nested blocks that specify such information as the product name, company name, copyrights, trademarks, operating system, and so on.

A language-specific block can contain any number of nested blocks. Each block corresponds to one of the language and character-set identifier pairs given in the resource's variable information block. The name of the language-specific block is a null-terminated string consisting of a concatenation of the language and character-set identifiers. The block has no corresponding value.

Each nested block contains a name that identifies version-specific information and a string that represents the value associated with the name. A nested block can have one of the following predefined names and associated values:

Name	Value
Comments	Specifies additional information that should be displayed for diagnostic purposes.
CompanyName	Specifies the company that produced the file--for example, "Microsoft Corporation" or "Standard Microsystems Corporation, Inc.". This string is required.
FileDescription	Specifies a file description to be presented to users. This string may be displayed in a list box when the user is choosing files to install--for example, "Keyboard Driver for AT-Style Keyboards" or "Microsoft Word for Windows". This string is required.
FileVersion	Specifies the version number of the file--for example, "3.10" or "5.00.RC2". This string is required.
InternalName	Specifies the internal name of the file, if one exists--for example, a module name if the file is a dynamic-link library. If the file has no internal name, this string should be the original filename, without extension. This string is required.
LegalCopyright	Specifies all copyright notices that apply to the file. This should include the full text of all notices, legal symbols, copyright dates, and so on--for example, "Copyright Microsoft Corporation 1990-1991". This string is optional.
LegalTrademarks	Specifies all trademarks and registered trademarks that apply to the file. This should include the full text of all notices, legal symbols, trademark numbers, and so on--for example, "Windows(TM) is a trademark of Microsoft Corporation". This string is optional.
OriginalFilename	Specifies the original name of the file, not including a path. This information enables an application to determine whether a file has been renamed by a user. The format of the name depends on the file system for which the file was created. This string is required.
PrivateBuild	Specifies information about a private version of the file--for example, "Built by TESTER1 on \TESTBED". This string should be present only if the <code>VS_FF_PRIVATEBUILD</code> flag is set in the <code>dwFileFlags</code> member of the <code>VS_FIXEDFILEINFO</code> structure of the root block.
ProductName	Specifies the name of the product with which the file is distributed--for example, "Microsoft Windows". This string is required.
ProductVersion	Specifies the version of the product with which the file is distributed--for example, "3.10" or "5.00.RC2". This string is required.
SpecialBuild	Specifies how this version of the file differs from the standard version--for example, "Private build for TESTER1 solving mouse problems on M250 and M250E computers". This string should be present only if the <code>VS_FF_SPECIALBUILD</code> flag is set in the <code>dwFileFlags</code> member of the <code>VS_FIXEDFILEINFO</code> structure in the root block.



## Symbol-File Format Overview (3.1)

This topic describes the format of symbol files created by Microsoft Symbol File Generator (MAPSYM). Symbol files contain information that the Microsoft Windows 80386 Debugger (WDEB386.EXE) can use to locate program modules and global data in an executable module.

The following topics describe the information in a symbol file:

Map Definitions  
Segment Definitions  
Symbol Definitions  
Constant Definitions  
Line Definitions

## Map Definitions

Every symbol file contains a list that links two or more map definitions. Each map definition describes a module in the executable file.

The first map definition in the chain starts at the beginning of the file, as follows:

```
/* File is loaded at pBuffer. */
```

```
pMapDef = (MAPDEF *)pFileBuffer;
```

Each map definition (except the last) contains a pointer to the next map definition in the chain. This pointer is a 16-bit number that, when multiplied by 16, gives the byte offset of the next map definition in the file, as follows:

```
pNextMapDef = (MAPDEF *) (pFileBuffer + (pMapDef->ppNextMap * 16));
```

The pointer in the last map definition is zero.

The MAPDEF structure for each map definition (except the last) has the following form:

```
typedef struct {
    WORD ppNextMap; /* paragraph pointer to next map */
    BYTE bFlags; /* symbol types */
    BYTE bReserved1; /* reserved */
    WORD pSegEntry; /* segment entry-point value */
    WORD cConsts; /* count of constants in map */
    WORD pConstDef; /* pointer to constant chain */
    WORD cSegs; /* count of segments in map */
    WORD ppSegDef; /* paragraph pointer to first segment */
    BYTE cbMaxSym; /* maximum symbol-name length */
    BYTE cbModName; /* length of module name */
    char achModName[1]; /* n bytes of module-name member */
} MAPDEF;
```

The last MAPDEF structure contains the version and release number for the version of Symbol File Generator used to create the symbol file. It has the following form:

```
typedef struct {
    WORD ppNextMap; /* always zero */
    BYTE release; /* release number (minor version number) */
    BYTE version; /* major version number */
} LAST_MAPDEF;
```

Following are the members of the MAPDEF structure:

ppNextMap	Specifies the offset from the beginning of the file to the next MAPDEF structure in the chain. Multiply the value of the ppNextMap member by 16 to obtain the offset.								
bFlags	Specifies the type of symbols in the file. The bFlags member can be one or more of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Contains 16-bit symbols.</td></tr><tr><td>1</td><td>Contains 32-bit symbols.</td></tr><tr><td>2</td><td>Includes alphabetic symbol table.</td></tr></tbody></table>	Value	Meaning	0	Contains 16-bit symbols.	1	Contains 32-bit symbols.	2	Includes alphabetic symbol table.
Value	Meaning								
0	Contains 16-bit symbols.								
1	Contains 32-bit symbols.								
2	Includes alphabetic symbol table.								
bReserved1	Reserved.								
pSegEntry	Specifies the segment of the entry point for the application.								
cConsts	Specifies the number of constants in this module.								
pConstDef	Specifies a 16-bit offset from the beginning of the file to an array of pointers to constant definitions. This value is not multiplied by 16 to obtain the offset.								
cSegs	Specifies the number of segments in this module.								
ppSegDef	Specifies the offset from the beginning of the file to the first segment definition in this module. Multiply the value of the ppSegDef member by 16 to obtain the offset.								
cbMaxSym	Specifies the length of the longest symbol name in this module.								
cbModName	Specifies the length of the module name.								

achModName Specifies a variable-length array of characters containing the module name. The name is not null-terminated.

## Segment Definitions

Each module in the symbol file contains a linked list of segment definitions. To obtain a pointer to the first segment definition, multiply the value of the ppSegDef member in the current MAPDEF structure by 16, as follows:

```
/* File is loaded at pFileBuffer. */  
  
pSegDef = (SEGDEF *) (pFileBuffer + (md.ppSegDef * 16));
```

Each segment definition contains a pointer to the next segment definition in the chain. This pointer is a 16-bit number that, when multiplied by 16, gives the byte offset of the next segment definition in the file, as follows:

```
pNextSegDef = (SEGDEF *) (pFileBuffer + (pSegDef->ppNextSeg * 16));
```

The pointer in the last segment definition is not zero. The linked list of segment definitions is circular--the pointer in the last segment definition gives the offset of the first segment definition. You can use the cSegs member in the MAPDEF structure to determine the number of segments in the module.

The SEGDEF structure for these lists has the following form:

```
typedef struct {  
    WORD ppNextSeg; /* paragraph pointer to next segment */  
    WORD cSymbols; /* count of symbols in list */  
    WORD pSymDef; /* offset of symbol chain*/  
    WORD wReserved1; /* reserved */  
    WORD wReserved2; /* reserved */  
    WORD wReserved3; /* reserved */  
    WORD wReserved4; /* reserved */  
    BYTE bFlags; /* symbol types*/  
    BYTE bReserved1; /* reserved */  
    WORD ppLineDef; /* offset of line-number record */  
    BYTE bReserved2; /* reserved */  
    BYTE bReserved3; /* reserved */  
    BYTE cbSegName; /* length of segment name*/  
    char achSegName[1]; /* n bytes of segment-name member */  
} SEGDEF;
```

Following are the members of the SEGDEF structure:

ppNextSeg	Specifies the offset from the beginning of the file to the next SEGDEF structure in the chain. Multiply the value of the ppNextSeg member by 16 to obtain the offset.								
cSymbols	Specifies the number of symbols in this segment.								
pSymDef	Specifies the offset from the beginning of the segment definition to an array of pointers to symbol definitions. This value is not multiplied by 16 to obtain the offset. For more details, see Section 12.3, " <a href="#">Symbol Definitions</a> ."								
wReserved1	Reserved.								
wReserved2	Reserved.								
wReserved3	Reserved.								
wReserved4	Reserved.								
bFlags	Specifies the type of symbols in this segment. The bFlags member can be one or more of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Contains 16-bit symbols.</td></tr><tr><td>1</td><td>Contains 32-bit symbols.</td></tr><tr><td>2</td><td>Includes alphabetic symbol table.</td></tr></tbody></table>	Value	Meaning	0	Contains 16-bit symbols.	1	Contains 32-bit symbols.	2	Includes alphabetic symbol table.
Value	Meaning								
0	Contains 16-bit symbols.								
1	Contains 32-bit symbols.								
2	Includes alphabetic symbol table.								
bReserved1	Reserved.								
ppLineDef	Specifies the offset from the beginning of the file to the first line-number definition. Multiply the value of the ppLineDef member by 16 to obtain the offset.								
bReserved2	Reserved.								
bReserved3	Reserved.								
cbSegName	Specifies the length of the segment name.								

achSegName      Specifies a variable-length array of characters containing the segment name. The name is not null-terminated.

## Symbol Definitions

Each segment definition contains a pointer to an array of pointers to symbol definitions.

All symbol files contain an array of pointers to symbols, sorted by symbol value. The `bFlags` member in the `SEGDEF` structure indicates whether the segment has an alphabetic symbol table. To obtain a pointer to the numerically ordered array of symbol-definition pointers, add the `pSymDef` pointer in the current segment definition to the pointer to the current segment definition, as follows:

```
aSymPtr = (WORD *)((BYTE *)pSegDef + pSegDef->pSymDef);
```

In addition, symbol files created by MAPSYM versions 5.0 and later may contain an array of pointers sorted alphabetically by symbol name. This array begins immediately after the numeric array:

```
aSymPtrAlpha = (WORD *)((BYTE *)pSegDef +  
    pSegDef->pSymDef + pSegDef->cSymbols * sizeof(WORD));
```

To obtain a pointer to each symbol definition, add the offset specified by each element in the array of symbol-definition pointers to the pointer to the current segment definition, as follows:

```
for (n = 0; n < pSegDef->cSymbols; n++) {  
    pSymDef = (SYMDEF *)((BYTE *)pSegDef + aSymPtr[n]);  
    .  
    .  
    /* Use the symbol information here. */  
    .  
    .  
}
```

The `SYMDEF` structure for these symbol definitions has the following form:

```
typedef struct {  
    WORD wSymVal; /* symbol address or constant */  
    BYTE cbSymName; /* length of symbol name */  
    char achSymName[1]; /* n bytes of symbol-name member */  
} SYMDEF;
```

Following are the members of the `SYMDEF` structure:

<code>wSymVal</code>	Specifies the address of the symbol or the value of a constant.
<code>cbSymName</code>	Specifies the length of the symbol name.
<code>achSymName</code>	Specifies a variable-length array of characters containing the segment name. The name is not null-terminated.

The `wSymVal` member in the `SYMDEF` structure is a doubleword value for 32-bit symbols.

## Constant Definitions

Each MAPDEF structure contains a pointer to an array of pointers to constant definitions. The format of a constant definition is the same as that of a symbol definition (you can use the SYMDEF structure described in Section 12.3, "Symbol Definitions").

The ppConstDef member in the current MAPDEF structure specifies the file offset of the array of constant-definition pointers, and the offset to each constant definition can be calculated from each element in the array, as follows:

```
aConstPtr = (WORD *) (pFileBuffer + md.ppConstDef);

for (n = 0; n < md.cConsts; n++) {
    pConstDef = (SYMDEF *) (pFileBuffer + aConstPtr[n]);
    .
    .
    /* Use the symbol information here. */
    .
    .
}
```

## Line Definitions

Symbol files created by linking with the `/LI` option also contain line-number information. Each segment definition contains a pointer to the first line definition in a circularly linked list. If the pointer in the `SEGDEF` structure is zero, the segment has no line-number information.

### LINEDEF Structure

To obtain a pointer to the first `LINEDEF` structure in the linked list, multiply the value of the `ppLineDef` member in the current `SEGDEF` structure by 16, as follows:

```
pLineDef = (LINEDEF *) (pBuf + (pSegDef->ppLineDef * 16));
```

Each `LINEDEF` structure (except the last) contains a pointer to the next `LINEDEF` structure in the linked list. The pointer in the last `LINEDEF` structure is zero.

The `LINEDEF` structure for each line definition has the following form:

```
typedef struct {
    WORD ppNextLine; /* ptr to next linedef (0 if last) */
    WORD wReserved1; /* reserved */
    WORD pLines; /* pointer to line numbers */
    WORD wReserved2; /* reserved */
    int cLines; /* count of line numbers */
    BYTE cbFileName; /* filename length */
    char achFileName[1]; /* filename (contains lines) */
} LINEDEF;
```

Following are the members of the `LINEDEF` structure:

<code>ppNextLine</code>	Specifies the offset from the beginning of the file to the next <code>LINEDEF</code> structure in the chain. Multiply the value of the <code>ppNextLine</code> member by 16 to obtain the offset. If this member is zero, there is no line-number information for this segment.
<code>wReserved1</code>	Reserved.
<code>pLines</code>	Specifies the offset from the beginning of the current <code>LINEDEF</code> structure to the array of line-information structures.
<code>wReserved2</code>	Reserved.
<code>cLines</code>	Specifies the number of lines in the line-information array.
<code>cbFileName</code>	Specifies the number of characters in the name of the source file. This file was compiled and linked to produce the map file.
<code>achFileName</code>	Specifies a variable-length array of characters containing the name of the source file. The name is not null-terminated.

### LINEINF Structure

To obtain a pointer to the first `LINEINF` structure in the array for the line-definition structure, add the `pLines` pointer in the current `LINEDEF` structure to the current `LINEDEF` pointer, as follows:

```
pLines = (LINEINF *) ((BYTE *) pLineDef + pLineDef->pLines);
```

Each element in the line-information array contains the offset into the source file for a line and the offset into the executable file for the code resulting from the source line.

The `LINEINF` structure has the following form:

```
typedef struct {
    WORD wCodeOffset; /* executable offset */
    WORD dwFileOffset; /* source offset */
} LINEINF;
```

Following are the members of the `LINEINF` structure:

<code>wCodeOffset</code>	Specifies the offset in this segment to the code resulting from compiling this line in the source file.
<code>dwFileOffset</code>	Specifies the offset to this line in the source file.



## Write File Format

This topic describes the binary file format used by Microsoft Write. A Write binary file contains information about file content, text and pictures (including object-linking-and-embedding, or OLE, objects), and formatting.

### Write-File Header

The Write-file header describes the content of the file. It contains data, pointers to subdivisions of the formatting section, and information about the length of the file. The file header has the following form:

Word	Name	Description
0	wIdent	Must be 0137061 octal (or 0137062 octal if the file contains OLE objects)
1	dy	Must be zero
2	wTool	Must be 0125400 octal
3		Reserved; must be zero
4		Reserved; must be zero
5		Reserved; must be zero
6		Reserved; must be zero
7-8	fcMac	Number of bytes of actual text plus 128, the bytes in one sector (low-order word first)
9	pnPara	Page number for start of paragraph information
10	pnFntb	Page number of footnote table (FNTB) or pnSep, if none
11	pnSep	Page number of section property (SEP) or pnSetb, if none
12	pnSetb	Page number of section table (SETB) or pnPgfb, if none
13	pnPgfb	Page number of page table (PGTB) or pnFfntb, if none
14	pnFfntb	Page number of font face-name table (FFNTB) or pnMac, if none
15-47	szSsht	Reserved for Microsoft Word compatibility
48	pnMac	Count of pages in whole file (last page number plus 1)

In the preceding list, a "page number" means an offset in 128-byte blocks from the start of the file. For example, if pnPara equals 10, the paragraph information is at offset  $10 * 128 = 1280$  in the file.

The starting page number of character information (pnChar) is not stored but is computable, as follows:

$$\text{pnChar} = (\text{fcMac} + 127) / 128$$

Examining the value of word 48 of the header is a good way to distinguish Write files from Microsoft Word files. If pnMac equals zero, the file originated in Word. Any other value identifies a Write file.

### Text and Pictures

After the header comes information about text and pictures. This information constitutes a separate section of the file.

#### Text

The text of the Write file starts at word 64 (page 1). Write uses the Windows character set (except for the pictures in the file) as well as the following special characters:

- ASCII character codes 13, 10 (carriage return, linefeed) for paragraph ends. No other occurrences of these two characters are allowed.
- ASCII character code 12 for explicit page breaks.
- ASCII character code 9 (normal) for tab characters.

Other line-break or wordwrap information is not stored.

#### Pictures

Pictures (including OLE objects) are stored as a sequence of bytes in the text stream. These bytes can be identified as picture information by examining their paragraph formatting. One picture is exactly one paragraph. Paragraphs that are pictures have a special bit set in their paragraph property (PAP) structure. For more information on the PAP structure, see Section 8.3, "Formatting."

Each picture consists of a descriptive header followed by the data that makes up the picture. The header for OLE objects is different from the one used for pictures. The picture header has the following form:

Byte	Name	Description
------	------	-------------

0-7	mfp	Windows <u>METAFILEPICT</u> structure (hMF member undefined)
8-9	dxaOffset	Offset of picture from left margin, in twips (1/1440 inch)
10-11	dxaSize	Horizontal size, in twips
12-13	dyaSize	Vertical size, in twips
14-15	cbOldSize	Number of following bytes (actual metafile or bitmap bits); set to zero
16-29	bm	Additional information for bitmaps only
30-31	cbHeader	Number of bytes in this header
32-35	cbSize	Number of following bytes (actual metafile or bitmap bits), replacing cbOldSize for new files
36-37	mx	Scaling factor (x)
38-39	my	Scaling factor (y)
40- ?	cbHeader	Picture contents, through cbHeader+cbSize- 1

The mm member (bytes 0-1) of the METAFILEPICT structure specifies the mapping mode used to draw the picture. The last set of bytes will be bitmap bits if the value of the mm member is 0xE3. This is a special value used only in Write. Otherwise, the bytes will be metafile contents.

If the picture has never been rescaled with the Size Picture command in Write, the scaling factors in each direction will be 1000 (decimal). If the picture has been resized, the scaling factor will be the percentage of the original size that the picture is now, relative to 1000 (100 per cent).

For information about the METAFILEPICT structure and bitmaps, see the Microsoft Windows Guide to Programming and the Microsoft Windows Programmer's Reference, Volumes 1 and 3.

The descriptive header for OLE objects is similar to the one used for pictures. The OLE object header has the following form:

Byte	Name	Description
0-1	mm	Must be 0xE4
2-5		Not used
6-7	objectType	Type: 1=static, 2=embedded, 3=link
8-9	dxaOffset	Offset of picture from left margin, in twips (1/1440 inch)
10-11	dxaSize	Horizontal size, in twips
12-13	dyaSize	Vertical size, in twips
14-15		Not used
16-19	dwDataSize	Number of bytes in the object data that follows the header
20-23		Not used
24-27	dwObjNum	Hexadecimal number that, when converted to an 8-digit string, represents the object's unique name
28-29		Not used
30-31	cbHeader	Number of bytes in this header
32-35		Not used
36-37	mx	Scaling factor (x)
38-39	my	Scaling factor (y)
40- ?	cbHeader	Object contents, through cbHeader+dwDataSize- 1

The scaling factors for OLE objects work the same way as they do with pictures.

### Formatting

Write files contain both character and paragraph formatting information. There can be no gaps in either; each must begin with the first text character (byte 128) and continue through the last. The format descriptors (FODs) for the first and last paragraph must, therefore, have the value of fcLim equal to the value of fcMac, as defined in the header section.

There is a difference between paragraph and character FODs. A character FOD may describe any number of consecutive characters with the same formatting. However, there must be exactly one paragraph FOD for each text paragraph. In either case, it is advisable to have multiple FODs point to the same formatting properties (FPROPs) on a given page because it saves space in the file. No FOD may point off its page.

### Characters and Paragraphs

Both the character and paragraph sections are structured as a set of pages. Each page contains an array of FODs and a group of FPROPs, both of which are described later in this section. Following is the format of a page:

Byte	Name	Description
0-3	fcFirst	Byte number of first character covered by this page of formatting information; equals 128 for first character in the text (low-order byte first)
4- n	rgfod	Array of FODs
n+1-126	grpfprow	Group of FPROPs
127	cfod	Number of FODs on this page

An FOD is fixed in size. It contains the byte offset to the corresponding FPROP. Following is the structure of an FOD:

Word	Name	Description
0-1	fcLim	Byte number after last character covered by this FOD
2	bfprop	Byte offset from beginning of FOD array to corresponding FPROP for these characters or this paragraph

An FPROP is variable in size. It contains the prefix for a character property (CHP) or paragraph property (PAP), both of which are described later in this section. Following is the structure of an FPROP:

Byte	Name	Description
0	cch	Number of bytes in this FPROP
1- n	rgchProp	Prefix for a CHP (for characters) or a PAP (for paragraphs) sufficient to include all bits that differ from the default CHP or PAP

Following is the format of a CHP:

Byte	Bit	Name	Description
0			Reserved; ignored by Write
1	0	fBold	Bold characters
	1	fItalic	Italic characters
	2-7	ftc	Font code (low bits); index into the FFNTB
2		hps	Size of font, in half points (standard is 24)
3	0	fUline	Underlined characters
	1	fStrike	Reserved; ignored by Write
	2	fDline	Reserved; ignored by Write
	3	fOverset	Reserved; ignored by Write
	4-5	csm	Reserved; ignored by Write
	6	fSpecial	Set for "(page)" only
	7		Reserved; ignored by Write
	4	0-2	ftcXtra
3		fOutline	Reserved; ignored by Write
4		fShadow	Reserved; ignored by Write
5-7			Reserved; ignored by Write
5		hpsPos	Position: 0=normal, 1-127=superscript, 128-255=subscript

If the user doesn't select any special character properties, the CHP is filled with the following default values:

Byte	Value
0	1
2	24
3-5	0

Each character FPROP must, therefore, have a count of characters (cch) greater than or equal to 1.

Each PAP can contain up to 14 tab descriptors (TBDs), which are described later in this section. Following is the structure of a PAP:

Byte	Bit	Name	Description
0			Reserved; must be zero
1	0-1	jc	Justification: 0=left, 1=center, 2=right, 3=both

2-7		Reserved; must be zero
2		Reserved; must be zero
3		Reserved; must be zero
4-5	dxaRight	Right indent, in 20ths of a point
6-7	dxaLeft	Left indent, in 20ths of a point
8-9	dxaLeft1	First-line left indent (relative to dxaLeft)
10-11	dyaLine	Interline spacing (standard is 240)
12-13	dyaBefore	Reserved; ignored by Write (standard is zero)
14-15	dyaAfter	Reserved; ignored by Write (standard is zero)
16	0	rhcPage 0=header, 1=footer
	1-2	Reserved; 0=normal paragraph, nonzero=header or footer paragraph
	3	rhcFirst Start of printing: 1=print on first page, 0=do not print on first page
	4	fGraphics Paragraph type: 1=picture, 0=text
	5-7	Reserved; must be zero
17-21		Reserved; must be zero
22-78		Tab descriptors (up to 14)

Following is the format of a TBD:

Byte	Bit	Name	Description
0-1		dxa	Indent from left margin of tab stop, in 20ths of a point
2	0-2	jcTab	Tab type: 0=normal tabs, 3=decimal tabs
	3-5	tlc	Reserved; ignored by Write
	6-7		Reserved; must be zero
3		chAlign	Reserved; ignored by Write

If the user doesn't select any special paragraph properties, the PAP is filled with the following default values:

Byte	Value
0	61
2	30
10-11	240 (word)
12-78	0

Each paragraph FPROP must have a count of characters (cch) greater than or equal to 1.

#### Footnotes

Write documents do not have footnote tables (FNTBs), so pnFntb is always equal to pnSep. In fact, all their header and footer paragraphs appear at the beginning of the document before any normal paragraphs. When reading files created by Word, Write recognizes only those headers and footers that appear at the beginning of the document; it treats all others as normal text.

#### Sections

A Write document has only one section. If the section properties of a Write document differ from the defaults, the document contains a section property (SEP) section and a section table (SETB) section. If not, then neither section is present and pnSep and pnSetb are both equal to pnPgtb.

Following is the format of an SEP:

Byte	Name	Description
0	cch	Count of bytes used, excluding this byte (all properties at byte positions greater than cch are set to their default values)
1-2		Reserved; must be zero
3-4	yaMac	Page length, in 20ths of a point (default is 11*1440=15840)
5-6	xaMac	Page width, in 20ths of a point (default is 8.5*1440=12240)
7-8		Reserved; must be 0xFFFF
9-10	yaTop	Top margin, in 20ths of a point (default is 1440)
11-12	dyaText	Height of text, in 20ths of a point (default is 9*1440=12960)
13-14	xaLeft	Left margin, in 20ths of a point (default is 1.25*1440=1800)

15-16 dxaText Width of text area, in 20ths of a point (default is 6\*1440=8640)

The page length (yaMac) is equal to yaTop+dyaText. The page width (xaMac) is equal to xaLeft+dxaText+(right margin, not stored).

If all the above properties are set to their defaults, no SEP or SETB is needed. Otherwise, the count of characters (cch) is greater than or equal to 1 and less than or equal to 16.

The SETB section contains an array of section descriptors (SEDs), described later in this section.

Following is the structure of an SETB:

Word	Name	Description
0	csed	Number of sections (always 2 for Write documents)
1	csedMax	Undefined
2- n	rgsed	Array of SEDs plus zero-padding to fill the sector

Following is the structure of an SED:

Word	Name	Description
0-1	cp	Byte address of first character following section
2	fn	Undefined
3-4	fcSep	Byte address of associated SEP

A Write document always has exactly two SED entries. The cp value of the first entry indicates that it affects all the characters in the document. The fcSep value of the first entry points to the one SEP in the file. The second SED entry is a dummy with fcSep set to 0xFFFFFFFF.

The PGTB section (optional) is on the page immediately after the SEP section.

Note: The term "page" used in the rest of this section refers to printed pages of a Write document, not 128-byte "pages" of a disk file.

The page table (PGTB) contains an array of page descriptors (PGDs), which are described later in this section. Following is the structure of a PGTB:

Word	Name	Description
0	cpgd	Number of PGDs (1 or more)
1	cpgdMac	Undefined
2- n	rgpgd	Array of PGDs plus zero padding to fill the sector

Following is the structure of a PGD:

Word	Name	Description
0	pgn	Page number in printed Word documents
1-2	cpMin	Byte address of first character on printed page

#### Font Table

The font face-name table (FFNTB) contains the number of font face names (FFNs) and a list of FFNs.

Following is the structure of an FFNTB:

Byte	Name	Description
0-1	cffn	Number of FFNs
2- n	grpffn	List of FFNs

Following is the structure of an FFN:

Byte	Name	Description
0-1	cbFfn	Number of bytes following in this FFN (not including these 2 bytes)
2	ffid	Font family identifier (see below)
3- (cbffn+2)	szFfn	Font name (variable length; null-terminated)

A cbffn value of 0xFFFF means that the next FFN entry will be found at the start of the next 128-byte page. A cbffn value of zero means that there are no more FFN entries in the table.

Possible values for ffid are FF\_DONTCARE, FF\_ROMAN, FF\_SWISS, FF\_MODERN, FF\_SCRIPT, and FF\_DECORATIVE. These constants are defined in WINDOWS.H. Additional values may be added to the list in future versions of Windows.

## Calendar File Format

This topic describes the binary file format used by Microsoft Windows Calendar (CALENDAR.EXE). A Calendar binary file contains information about file content, dates, days, and appointments.

### Calendar-File Header

The first 8 bytes of a Calendar file are a character array identifying the file as a Calendar file. Following are the contents of the array:

```
'C' + 'r' = b5
'A' + 'a' = a2
'L' + 'd' = b0
'E' + 'n' = b3
'N' + 'e' = b3
'D' + 'l' = b0
'A' + 'a' = a2
'R' + 'c' = b5
```

The next 2 bytes (cDateDescriptors) contain the integer count of dates described in the file.

The next 12 bytes contain six 2-byte fields of information that is global to the entire file. These variables are normally set by the user through the Alarm Controls and Options Day dialog boxes. The header information has the following form:

```
WORD    MinEarlyRing
BOOL    fSound
int     interval
int     mininterval
BOOL    f24HourFormat
int     StartTime
```

Following are the members in the header structure:

MinEarlyRing	Specifies an early ring, in minutes.
fSound	Specifies whether alarms should be audible.
interval	Specifies the interval between appointments: 0 = 15 minutes, 1 = 30 minutes, 2 = 60 minutes.
mininterval	Specifies the interval, in minutes.
f24HourFormat	Specifies the time format: nonzero=24-hour format.
StartTime	Specifies the starting time in day mode--that is, the time that normally appears first in the display, in minutes past midnight.

The rest of the first 64 bytes are reserved.

### Date Descriptors

A date-descriptor array appears next. Each entry in the array describes one day. The number of entries in the array is cDateDescriptors (described in the preceding section). Each element in the array consists of 12 bytes, in six 2-byte fields. The date-descriptor array has the following form:

```
unsigned    Date
int         fMarked
int         cAlarms
unsigned    FileBlockOffset
int         reserved
unsigned    reserved
```

Following are the members in the date-descriptor array:

Date	Specifies the date, in days past 1/1/1980.
fMarked	Specifies which mark(s) are set for the date: box = 128, parentheses = 256, circle = 512, cross = 1024, underscore = 2048.
cAlarms	Specifies the number of alarms set for the day.
FileBlockOffset	Specifies the file offset, in 64-byte blocks, to the day's information. Only the low 15 bits are used (the high bit will be zero). Thus, if this offset is 6, the day's information is stored at byte 6*64 in the file.
reserved	Reserved; must be 0xFFFF.

reserved                      Reserved; must be 0xFFFF.

#### Day-Specific Information

All day information is stored after the date-descriptor array, on even 64-byte boundaries. The day-information structure has the following form:

```
unsigned     reserved
unsigned     Date
unsigned     reserved
unsigned     cbNotes
unsigned     cbAppointment
char        Notes[cbNotes]
BYTE        ApptInfo[ ]
```

Following are the members in the day-information structure:

reserved	Reserved; must be zero.
Date	Specifies the date, in days past 1/1/1980.
reserved	Reserved; must be 1.
cbNotes	Specifies the number of bytes of note information, including null bytes. This information appears in the note array below the appointment list.
cbAppointment	Specifies the count of bytes of appointment information.
Notes	Contains the text of the note.
ApptInfo	Contains the block of appointments.

#### Appointment-Specific Information

The information in the appointment block is stored as a list of single appointments. Each appointment consists of a structure similar to the following:

```
struct {
    char cb;
    char flags;
    int time;
    char szApptDesc[ ];
};
```

Following are the members in each appointment structure:

cb	Specifies the size, in bytes, of the structure containing the appointment. The structure address of the next appointment is the current appointment plus the value of the cb member.						
flags	Contains various flags. This member can have one or more of the following values: <table><tr><th>Value</th><th>Meaning</th></tr><tr><td>1</td><td>Alarm will go off at the specified time of the appointment.</td></tr><tr><td>2</td><td>Appointment is a special time.</td></tr></table>	Value	Meaning	1	Alarm will go off at the specified time of the appointment.	2	Appointment is a special time.
Value	Meaning						
1	Alarm will go off at the specified time of the appointment.						
2	Appointment is a special time.						
time	Specifies the number of minutes past midnight.						
szApptDesc	Contains a null-terminated string consisting of text associated with an appointment.						

## Installable Drivers

This topic describes installable drivers and the installable-driver interface for the Microsoft Windows operating system. Topics discussed in this topic include: the common entry point for installable drivers, messages used by the common entry point, actions that an installable driver should take in response to these messages, and functions available for the installable driver interface.

### About Installable Drivers

An installable driver is a Windows dynamic-link library (DLL) that a Windows application (or another Windows DLL) can open, enable, query, disable, and close. An application can perform these operations by calling the following functions:

Function	Description
<u>CloseDriver</u>	Closes an installable driver.
<u>GetDriverInfo</u>	Retrieves installable-driver data.
<u>GetDriverModuleHandle</u>	Retrieves an installable driver's module handle.
<u>GetNextDriver</u>	Enumerates installed drivers.
<u>OpenDriver</u>	Opens an installable driver.
<u>SendDriverMessage</u>	Sends a message to an installable driver.

When an application calls the OpenDriver, SendDriverMessage, or CloseDriver function, Windows processes the call and issues one or more of the following driver messages:

Message	Description
DRV_CLOSE	Notifies an installable driver that Windows will decrement the use count for the driver and send a <u>DRV_FREE</u> message if the use count reaches zero.
DRV_CONFIGURE	Notifies an installable driver that it should display a custom-configuration dialog box. (This message should only be sent if the driver returns a nonzero value when the <u>DRV_QUERYCONFIGURE</u> message is processed.)
DRV_DISABLE	Notifies an installable driver that the memory that it has allocated is about to be freed.
DRV_ENABLE	Notifies an installable driver that it has been loaded or reloaded or that Windows has been enabled.
DRV_FREE	Notifies an installable driver that it will be discarded.
DRV_INSTALL	Notifies an installable driver that it has been successfully installed.
DRV_LOAD	Notifies an installable driver that it has been successfully loaded.
DRV_OPEN	Notifies an installable driver that it is about to be opened.
DRV_POWER	Notifies an installable driver that the power source for the device is about to be turned off or on.
DRV_QUERYCONFIGURE	Queries an installable driver about whether it supports the <u>DRV_CONFIGURE</u> message and can display a private configuration dialog box.
DRV_REMOVE	Notifies an installable driver that it is about to be removed from the system.

These messages, which are defined in the Windows header file (WINDOWS.H), are processed by the main routine in an installable driver. This routine is called the DriverProc function.

Some of the preceding messages should be sent by Windows only when one of the installable driver functions is called by an application. The circumstances under which these messages are sent are described in the following list:

Message	Description
DRV_CLOSE	Issued by Windows when an application calls the <u>CloseDriver</u> function.
DRV_DISABLE	Issued prior to exiting Windows and returning to MS-DOS or when the driver is freed.
DRV_ENABLE	Issued when returning to Windows from MS-DOS or the first time the installable driver is loaded.
DRV_FREE	Issued by Windows after an application calls the <u>CloseDriver</u> function and the use count is decremented to zero.



**DRV\_LOAD** Issued by Windows after the first OpenDriver call is made for a particular installable driver.

The remaining messages can be sent by an application to an installable driver by calling the SendDriverMessage function.

#### Creating an Installable Driver

An installable driver is a Windows dynamic-link library (DLL) that supports a special entry point, the DriverProc function. This function processes the driver messages described in the previous section. This function may also process private driver messages. These messages can be assigned values ranging from DRV\_RESERVED to DRV\_USER (two constants that appear in WINDOWS.H).

The following example shows the basic structure of the DriverProc function:

```
LRESULT CALLBACK* DriverProc (DWORD    dwDriverIdentifier,
HDRVR    hDriver,
UINT    wMessage,
LPARAM    lParam1,
LPARAM    lParam2)
{
    DWORD dwRes = 0L;

    switch (wMessage)
    {

        case DRV_LOAD:

            /* Sent when the driver is loaded. This is always */
            /* the first message received by a driver.*/

            dwRes = 1L; /* Return 0L to fail.*/
            break;

        case DRV_FREE:

            /* Sent when the driver is about to be discarded. */
            /* This is the last message a driver receives */
            /* before it is freed.*/

            dwRes = 1L; /* Return value is ignored. */
            break;

        case DRV_OPEN:

            /* Sent when the driver is opened. */

            dwRes = 1L; /* Return 0L to fail.*/
            /* This value is subsequently used */
            /* for dwDriverIdentifier.*/

            break;

        case DRV_CLOSE:

            /* Sent when the driver is closed. Drivers are */
            /* unloaded when the open count reaches zero. */

            dwRes = 1L; /* Return 0L to fail.*/
            break;

        case DRV_ENABLE:

            /* Sent when the driver is loaded or reloaded and */
            /* when Windows is enabled. Hook or rehook*/
            /* interrupts and initialize hardware. Expect the */
    }
```

```

/* driver to be in memory only between the enable */
/* and disable messages. */

dwRes = 1L; /* Return value is ignored. */
break;

    case DRV_DISABLE:

/* Sent before the driver is freed or when Windows */
/* is disabled. Unhook interrupts and place */
/* peripherals in an inactive state. */

dwRes = 1L; /* Return value is ignored. */
break;

    case DRV_INSTALL:

/* Sent when the driver is installed.*/

dwRes = DRV_OK; /* Can also return DRV_CANCEL */
/* and DRV_RESTART. */
break;

    case DRV_REMOVE:

/* Sent when the driver is removed. */

dwRes = 1L; /* Return value is ignored. */
break;

    case DRV_QUERYCONFIGURE:

/* Sent to determine if the driver can be */
/* configured. */

dwRes = 0L; /* Zero indicates configuration*/
/* NOT supported. */
break;

    case DRV_CONFIGURE:

/* Sent to display the custom-configuration */
/* dialog box for the driver. */

dwRes = DRV_OK; /* Can also return DRV_CANCEL */
/* and DRV_RESTART. */
break;

    default:

/* Process any messages not explicitly trapped.*/

return DefDriverProc (dwDriverIdentifier, hDriver,
    wMessage, lParam1, lParam2);
}
return dwRes;
}

```

#### Opening an Installable Driver

An application opens an installable driver by calling the OpenDriver function. When an application calls this function, Windows adds the driver name to an internal list of installed drivers. (When the application calls the CloseDriver function, Windows deletes the corresponding driver name from this list.)

When an application calls the OpenDriver function to open the first instance of a driver, Windows issues the DRV\_LOAD, DRV\_ENABLE, and DRV\_OPEN messages, in that order. (Subsequent calls to OpenDriver cause only DRV\_OPEN to be sent.) When the driver processes the DRV\_LOAD message, it reads the configuration settings (if any exist) from the corresponding entry in the SYSTEM.INI file and configures the driver and any associated hardware. In addition to configuring the driver and associated hardware, the driver also allocates required memory.

After processing the DRV\_LOAD message, the driver returns a nonzero value if it loads successfully. If it returns zero, Windows immediately unloads the driver (without issuing a DRV\_FREE message).

When the driver processes the DRV\_ENABLE message, it hooks or chains required interrupts and prepares associated peripherals.

When the driver processes the DRV\_OPEN message, it allocates memory required by a single instance of the driver.

#### Closing an Installable Driver

An application closes an installable driver by calling the CloseDriver function. When the application calls this function, Windows deletes the corresponding driver name from an internal list.

When an application calls the CloseDriver function to close the last instance of a driver, Windows issues the DRV\_CLOSE, DRV\_DISABLE, and DRV\_FREE messages, in that order. (When the application is not closing the last instance of the driver, only DRV\_CLOSE is sent.) When the driver processes the DRV\_CLOSE message, it frees any resources that were allocated when the driver was opened and returns a nonzero value. If the driver returns a value of zero, closing fails.

When the driver processes the DRV\_DISABLE message, it places any associated peripherals in an inactive state and unhooks all interrupts.

When the driver processes the DRV\_FREE message, it frees any resources that are still allocated.

#### Configuring an Installable Driver

Many installable drivers support a private configuration dialog box that lets the user configure the driver and associated hardware. To determine whether a driver supports such a dialog box, an application calls the SendDriverMessage function and issues the DRV\_QUERYCONFIGURE message. If the driver is configurable, this function returns a nonzero value. If it is not configurable, this function returns zero. If the SendDriverMessage function returns a nonzero value, the application displays the configuration dialog box by calling the SendDriverMessage function a second time and sending the DRV\_CONFIGURE message.

If the driver supports a private configuration dialog box, it should display the dialog box and process user input when it receives the DRV\_CONFIGURE message. Typically, any configuration data specified by the user is maintained in the [drivers] section of the Windows SYSTEM.INI file.

#### Enumerating Instances of an Installable Driver

An application can retrieve a handle identifying either the first instance of an installable driver or each instance of the driver by calling the GetNextDriver function.

#### Updating the SYSTEM.INI File

Upon installation, the [drivers] section of the SYSTEM.INI file contains an entry for each installable driver. This entry has the following form:

entry=driver\_filename optional\_information

An application can open a driver by using its filename or its entry. If a fully qualified path is not specified with the filename, the driver file must exist on the standard Windows search path. The driver interface searches for the driver as follows:

- If an application specifies a section name, that section of SYSTEM.INI is searched instead of the [drivers] section.
- If an application specifies an entry in the search section, the driver with a filename corresponding to the entry is opened.
- If the string specified by the application does not match an entry in the search section, the system assumes the string is a driver filename.

The optional information (optional\_information) following the driver name (driver\_filename) lists information a driver needs after installation. A driver maintains configuration information here if the information is limited or if it needs to be associated with the entry. For example, two prototype drivers could be installed in the system. The first driver could be associated with serial port one, and the second

driver could be associated with serial port two. The [drivers] section of the SYSTEM.INI might show this association in the following way:

```
[drivers]
prototype1=proto.driv com1
prototype2=proto.driv com2
```

If your driver uses more extensive configuration information, it can create a section in the SYSTEM.INI file reserved for its parameters. For example, the installable driver PROTO.DRV might create the following [proto.driv] section:

```
[proto.driv]
port=230
int=3
```

When reserving a section for your driver, use the filename of your driver to identify the section. A driver usually configures and maintains this section of information when it displays the configuration dialog box used for the DRV\_CONFIGURE message.

If you want your installable driver loaded when Windows starts, place its filename or an alias from the [drivers] section of the SYSTEM.INI file on the drivers= line of the [boot] section found in the SYSTEM.INI file. Windows loads these drivers at startup and sends DRV\_LOAD and DRV\_ENABLE messages to them but does not open them. This makes it possible for you to install drivers that remain resident while Windows is enabled.

#### Contents of the OEMSETUP.INF Files

The OEMSETUP.INF file uses the same format as the Windows 3.0 SETUP.INF file with the exception of a new [Installable.Drivers] section. This section identifies the names and characteristics of each driver on the disk. Each driver entry has the following form:

```
entry=disk:filename, type(s), description, VxD(s), default_params
```

Note that the elements that compose a driver entry are separated by commas. Comments are delimited by semicolons; all characters following a semicolon are considered part of the comment string.

Following are the elements that compose a driver entry:

Element	Description
entry	Identifies the driver. This string must be unique.
disk	Specifies the disk number for the disk that contains the driver. This entry corresponds to an entry in the [disks] section of SETUP.INF.
filename	Specifies the name of the file that contains the driver.
type(s)	Specifies the driver type.
description	Describes the driver. This string appears in the dialog box displayed by the Drivers Control Panel application.
VxDs	Identifies any VxDs required by the driver. (For a description of the manner in which multiple VxD names are parsed, see the Microsoft Windows Virtual Device Adaptation Guide.)
default_params	Specifies default parameters for the driver. Additional options are appended to the driver entry in the [drivers] section of SYSTEM.INI.

If you create an OEMSETUP.INF file to distribute with your driver, it must include the [disks] and [Installable.Drivers] sections. For example, the following entries could be used in an OEMSETUP.INF file for a prototype installable driver:

```
[disks]
; Numeric mappings for disk titles

1 = ., "Sample Distribution Disk 1"

[Installable.Drivers]
; The installable drivers section is unique to the drivers application.
; It is parsed with comma-separated fields.

prototype=1:proto.driv,"ampl,freq","Sample scope driver","1:VXDA.386"
```

The Drivers Control Panel application may need to copy files that support your driver. If any of these files are not VxDs, include a section in the SYSTEM.INI file listing them. Use the entry (that is, prototype) as the name of this new section. For example, if the prototype driver has an additional file called POWERSRC.DLL, include the following section:

```
[prototype]
; Keyname sections can be created for dependent files. All
; dependent files will be copied directly to the system directory.

1:POWERSRC.DLL
```

#### Drivers Control Panel Application

The Drivers Control Panel application installs, modifies, and removes drivers. When started, the Drivers Control Panel application displays the following dialog box. The application searches the installed Drivers list box displays the description strings of the installed drivers, the installed drivers and loading them. If a description string does not match an installed driver, the application searches the MMSETUP.INF file and then the header of the driver file to obtain the description string. A scroll bar appears in the list box if there are more drivers than can be displayed.

The following buttons are found in the Control Panel dialog box:

Button	Result when chosen
OK	Exits the dialog box and makes any changes permanent.
Cancel	Exits the dialog box. The application ignores any requests to install or remove drivers made during the session. Any configuration changes made during the session are retained because they are done by the driver.
Remove	Removes the information about the selected driver from the SYSTEM.INI file. When removing drivers, the Control Panel application sends the <u>DRV_REMOVE</u> message to the driver if there is only one entry in the SYSTEM.INI file for it.
Setup	Applies only to configurable drivers. When the user selects a driver in the list box, the application opens the driver and sends it the <u>DRV_QUERYCONFIGURE</u> message. If a driver responds that it can be configured--that is, it supports a configuration dialog box to set such parameters as the COM port, the interrupt number, or input and output (I/O) port address--then the application enables the Setup button. If the user chooses the Setup button, the application sends a <u>DRV_CONFIGURE</u> message to the driver.
Add Drivers	Installs a new driver.
Default	Redisplays the list of files from the MMSETUP.INF file. Note that the Default button is active when the OEM drivers are displayed.

#### Installing a Driver

When the user selects a driver from the Installed Drivers list box, the Add Driver dialog box closes. The new driver becomes selected in the list box when the user chooses the OK button. The Drivers Control Panel application sends the DRV\_INSTALL message to the driver if there is only one entry in the SYSTEM.INI file for it. (A driver receives the DRV\_INSTALL message for its initial installation.) The Drivers Control Panel application can install up to four wave devices, four musical instrument digital interface (MIDI) devices, and ten media control interface (MCI) devices of the same type.

If the selected driver is not an installable driver, the Driver Control Panel applications displays a "Cannot Install" message. If the user chooses the Cancel button, the dialog box closes with no changes made.

#### Using Drivers with the Drivers Control Panel Application

During installation, the Drivers Control Panel application opens the driver and obtains the description line, originally defined in the module-definition (.DEF) file, from the driver header. The application uses the description line to construct the settings for the [drivers] section. The description line in the .DEF file should have the following form:

```
DESCRIPTION type(s):text
```

Following are the parameters in the description line:

Parameter	Meaning
type(s)	Type of driver used for the entry in the SYSTEM.INI file. Multiple entries are separated by commas.
text	Text that describes the driver. This will be displayed in the Drivers Control Panel application.

For example, the header file for an oscilloscope driver (OSCI.DRV) can use the following description line:

```
DESCRIPTION 'FREQ,AMPL:Oscilloscope frequency and amplitude drivers.'
```

Based on this definition, if both drivers are installed (that is, if the Drivers Control Panel application displays a selection for both FREQ and AMPL), the Drivers Control Panel application creates the following settings in the SYSTEM.INI file:

```
[drivers]
FREQ = osci.drv
AMPL = osci.drv
```

If you want your driver added to a named section of the SYSTEM.INI file, you can add the section name to the type of driver. For example, the following description line specifies that a voltmeter driver be added to the [RCC] section:

```
DESCRIPTION 'VOLTMETER[RCC]:RCC voltmeter driver.'
```

#### Creating a Custom Configuration Application

The Drivers Control Panel application provides a convenient interface for installing drivers. You should use this interface for configuring features that are hardware- or driver-dependent.

If your driver configures system features--those features that are hardware- and device-independent--you should create a custom Control Panel application.

## BootApp (3.1)

```
void BootApp(hBlock, hFile)
HANDLE hBlock;      /* handle of information block */
HANDLE hFile;       /* handle of executable file */
file
/
```

The BootApp function loads the given application.

Parameter	Description
hBlock	Identifies the selector for the segment that contains the information block in the Windows (new-style) header.
hFile	Identifies the executable file that contains the application. The hFile parameter must be a valid MS-DOS file handle.

### Returns

This function does not return a value.

### Comments

The information block in the Windows header that is identified by the hBlock parameter specifies the linker version number, the length of various tables of data, offsets to those tables, heap and stack sizes, and so on.

The BootApp function is one of three functions required for self-loading Windows applications. The application developer must provide the code for this function and store a pointer to the function at offset 0x0004 in the application's loader code and data table.

The Windows kernel calls this function after loading the application's executable header and data tables.

### EntryAddrProc (3.1)

```
DWORD EntryAddrProc(hBlock, wEntryNo)
HANDLE hBlock;      /* selector for information block */
WORD wEntryNo;      /* entry-table
procedure index
/
```

The EntryAddrProc function retrieves an address for the specified procedure.

Parameter	Description
hBlock	Specifies the selector for the segment that contains the information block in the Windows (new-style) header.
wEntryNo	Specifies the index to the entry in an entry table that identifies the procedure for which the function should return an address.

#### Returns

The return value is the address of the specified procedure if the function is successful. Otherwise, the return value is zero.

#### Comments

The wEntryNo parameter is also known as the procedure's ordinal number.

The EntryAddrProc function is one of three functions supplied by the Windows kernel. The kernel loads a pointer to this function at offset 0x0014 in the loader's code and data table. The kernel loads the pointer before calling the private startup procedure (the BootApp function).

EntryAddrProc is called from the LoadAppSeg function, which the application developer must supply.



## ExitProc (3.1)

```
void ExitProc(hBlock)  
HANDLE hBlock;      /* selector of information block */
```

The ExitProc function closes a self-loading application.

Parameter	Description
hBlock	Specifies the selector for the segment that contains the information block in the Windows (new-style) header.

### Returns

This function does not return a value.

### Comments

The Windows header information block identified by the hBlock parameter specifies the linker version number, the length of various tables of data, offsets to those tables, heap and stack sizes, and so on.

The ExitProc function is one of three functions required for self-loading Windows applications. The application developer must provide the code for this function and store a pointer to it at offset 0x0018 in the application's loader code and data table.

ExitProc does not need to free memory owned by the application, nor is it necessary for the function to close any open files.

### MyAlloc (3.1)

```
DWORD MyAlloc(wFlags, wSize, wElem)
WORD wFlags;      /* segment flags          */
WORD wSize;       /* size of element */
WORD wElem;
* number of elements in segment
/
```

The MyAlloc function allocates memory for a segment in a self-loading application.

Parameter	Description
wFlags	Specifies the segment flags.
wSize	Specifies the element size, in bytes.
wElem	Specifies the number of elements in the segment.

#### Returns

The low-order word of the return value contains a segment handle if the function is successful; the high-order word contains a selector if the function is successful. (However, if the function allocates only a handle for the segment, the low-order word contains zero and the high-order word contains the handle.) Otherwise, the return value is zero for both high-order and low-order words.

#### Comments

The flags specified by the wFlags parameter are the values that precede the segment table appearing immediately after the information block in the Windows (new-style) header. The kernel translates wFlags into the proper values before calling the GlobalAlloc function.

The segment size, in bytes, is obtained by shifting the value specified in the wSize parameter left by the number of bits specified by the wElem parameter.

The MyAlloc function is one of three functions supplied by the Windows kernel. The kernel loads a pointer to this function at offset 0x0014 in the loader's code and data table. The kernel loads the pointer before calling the private startup procedure (the BootApp function).

#### See Also

BootApp

## PatchCodeHandle (3.1)

```
void PatchCodeHandle(hSeg)
WORD hSeg;    /* segment with entry points */
```

The PatchCodeHandle function modifies prolog code for self-loading applications.

Parameter	Description
hSeg	Identifies the segment containing the entry points to be modified.

### Returns

This function does not return a value.

### Comments

This function is one of four supplied by the Windows kernel. An application can reference this function by including the following statement in the IMPORTS section of its module-definition file:

```
PatchCodeHandle = KERNEL.110
```

If the entry points in a segment use the DS register, the prolog is modified as follows:

Original prolog	Modified prolog
push ds	mov ax, dgroup
pop ax	
nop	

If the entry points do not use the DS register, the prolog is modified as follows:

Original prolog	Modified prolog
push ds	mov ax, ds
pop ax	nop
nop	

The loader calls the PatchCodeHandle function from the LoadAppSeg function, which reloads a segment that has been discarded. The loader must call the SetOwner function and identify the segment's owner before it calls the PatchCodeHandle function. In addition, the loader must set the SINGLEDATA bit in the Windows (new-style) header's information block.

### See Also

LoadAppSeg, SetOwner

### LoadAppSeg (3.1)

```
WORD LoadAppSeg(hBlock, hFile, wSegID)
HANDLE hBlock;      /* handle of module information block */
HANDLE hFile;       /* handle of
executable file
/
WORD wSegID;
* segment identifier
/
```

The LoadAppSeg function loads a segment for the first time or reloads a discarded segment. The segment is identified by the wSegID parameter and belongs to the given application.

Parameter	Description
hBlock	Specifies the segment selector for the segment containing the module information block.
hFile	Identifies the executable file that contains the application. This parameter is an MS-DOS file handle. (This handle is -1 if the file is not open.)
wSegID	Identifies the segment that the function should reload.

#### Returns

The return value is a selector for the segment if the function is successful. Otherwise, it is zero.

#### Comments

The information block in the Windows (new-style) header identified by the hBlock parameter specifies the linker version number, the length of various tables of data, offsets to those tables, heap and stack sizes, and so on.

The third parameter, wSegID, is determined by the linker at link time.

The LoadAppSeg function is one of three functions required for self-loading Windows applications. The application developer must provide the code for this function and store a pointer to it at offset 0x0008 in the application's loader code and data table.

### SetOwner (3.1)

```
void SetOwner(hSel, hOwner)
WORD hSel;          /* selector of segment */
HANDLE hOwner;      /* handle of
information block */
/
```

The SetOwner function associates the given segment with an executable file or application.

Parameter	Description
hSel	Specifies a selector or handle identifying the segment to be associated with the executable file or application.
hOwner	Identifies the information block in the Windows (new-style) executable-file header for the application that contains the segment.

#### Returns

This function does not return a value.

#### Comments

The Windows header information block identified by the hOwner parameter specifies the linker version number, the length of various tables of data, offsets to these tables, heap and stack sizes, and so on.

The SetOwner function is one of three functions required for self-loading Windows applications. The application developer must provide the code for this function and store a pointer to it at offset 0x0004 in the application's loader code and data table.

After the kernel allocates memory for a segment by using the MyAlloc function, it calls SetOwner.

## Floating Point Emulation Library

This topic describes two methods that can be used to support floating-point emulation in Windows applications. In particular, the topic describes in detail the Windows 80x87 floating-point emulator in the dynamic-link library WIN87EM.DLL. This information is intended to be used by compiler vendors who want to develop floating-point emulators that are compatible with WIN87EM.DLL.

### Emulation Methods

With floating-point emulation, Windows applications that contain floating-point instructions can run on any computer, regardless of whether the computer has floating-point hardware.

To support floating-point emulation for Windows applications, compiler vendors can use one of the following methods:

- Emulation by exception handler
- Windows 80x87 floating-point emulation

### Emulation by Exception Handler

With emulation by exception handler, a Windows application contains floating-point instructions for all floating-point operations and an exception handler for occurrences of Interrupt 07h (coprocessor not available). When the application starts, it installs the exception handler and the exception handler processes any floating-point exceptions that occur thereafter.

When the application runs on a computer with no floating-point hardware, a floating-point exception occurs the first time a floating-point instruction is executed. The exception handler is responsible for patching and then restarting the instruction. To patch the floating-point instruction, the exception handler actually replaces it with a call to emulation code. The new instruction calls the emulation code directly (rather than generating an exception) for as long as the patched instruction remains in memory.

This method can be used only with the Microsoft Windows operating system, version 3.1, because Windows version 3.0 standard mode does not save and restore the state of the exception handler across task switches.

This method may be less efficient than other methods because it requires that floating-point instructions be patched while the application is running rather than while it is loading. As long as the patched instructions remain in memory, however, this method is as efficient as other methods. If Windows discards the code segments that contain the patched instructions, the floating-point instructions must be patched again because Windows always loads a fresh copy of the code when it restores the discarded segments.

### Windows 80x87 Floating-Point Emulation

With Windows 80x87 floating-point emulation, the Windows application contains calls to floating-point instructions for all floating-point operations, but the application also includes fixup records for each instruction. When Windows loads the application, Windows determines whether floating-point hardware is present. If the hardware is not present, Windows uses the fixup records to replace the actual instructions with calls to emulation code.

To support this method, the application's startup routine must check whether WIN87EM.DLL is present. Then the routine must initialize WIN87EM.DLL by calling the `__fpmath` function with the BX register set to 0 and must set the floating-point exception handler by calling the `__fpmath` function with the BX register set to 3 and the DS:AX registers pointing to the exception handler. When the application's `WinMain` function returns to the startup routine, the routine must release WIN87EM.DLL by calling the `__fpmath` function with the BX register set to 2. After WIN87EM.DLL has been released, the startup routine can end the application.

For this method to work correctly, the Windows application must contain the proper fixup records--sometimes called operating system (OS) fixups--to convert instructions to emulation calls. For WIN87EM.DLL, each call consists of an interrupt (int) instruction followed by one or more words defining the floating-point operation and operands. The call is actually generated by the addition of fixup values to the first two words of the corresponding floating-point instruction. The fixup values to use depend on the instruction--the values are defined as follows:

```
fINT      equ0CDh
fFWAIT    equ09Bh
fESCAPE   equ0D8h
fFNOP     equ090h
fESequ026h
fCSequ02Eh
```

```

fSsequ036h
fDsequ03Eh
BEGINT equ034h

FIARQQ equ(fINT + 256*(BEGINT + 8)) - (fFWAIT + 256*fDS)
FISRQQ equ(fINT + 256*(BEGINT + 8)) - (fFWAIT + 256*fSS)
FICRQQ equ(fINT + 256*(BEGINT + 8)) - (fFWAIT + 256*fCS)
FIERQQ equ(fINT + 256*(BEGINT + 8)) - (fFWAIT + 256*fES)
FIDRQQ equ(fINT + 256*(BEGINT + 0)) - (fFWAIT + 256*fESCAPE)
FIWRQQ equ(fINT + 256*(BEGINT + 9)) - (fFNOP + 256*fFWAIT)
FJARQQ equ256*(((0 shl 6) or (fESCAPE and 03Fh)) - fESCAPE)
FJSRQQ equ256*(((1 shl 6) or (fESCAPE and 03Fh)) - fESCAPE)
FJCRQQ equ256*(((2 shl 6) or (fESCAPE and 03Fh)) - fESCAPE)

```

Each of the six fixup record types consists of two one-word values, as shown in the following example:

```

osfixuptbl label word
    DW FIARQQ, FJARQQ
    DW FISRQQ, FJSRQQ
    DW FICRQQ, FJCRQQ
    DW FIERQQ, 0h
    DW FIDRQQ, 0h
    DW FIWRQQ, 0h
osfixuptbl len = $-osfixuptbl

```

The loader assumes that each floating-point instruction is preceded by a wait instruction. The loader adds the first word to the combination of the wait instruction byte and the first byte in the floating-point instruction. For fixup types 1 through 3, the loader adds the second word to the second and third bytes of the floating-point instruction. For types 4 through 6, the loader makes no changes to these bytes (it adds zero).

Because WIN87EM.DLL polls for exceptions by using the fwait instruction, the loader must replace each nop and fwait instruction pair with a call to emulation code, even if a floating-point coprocessor is available. These instructions must have a corresponding fixup record of type 6.

WIN87EM.DLL does not emulate the following floating-point instructions:

```

fbld      fsave
fbstp     fsetpm
fcos      fsin
fdecstp   fsincos
fincstp   fstenv
finit     fucom
fldenv    fucomp
fnop      fucompp
fprem1    fextract
frstor

```

### Windows 3.0 Limitations

Windows 3.0 does not correctly save and restore the emulator state for emulator functions 0x38 through 0x3E. This means that Windows applications that use a floating-point emulator other than WIN87EM.DLL may not run successfully if another application that is using WIN87EM.DLL is also running.

Windows 3.1 does correctly save and restore the emulator state. Therefore, applications that use other floating-point emulators should be run only under Windows 3.1.

## `__fpmath (2.x)`

`extern __fpmath:far`

```
mov  bx, Function ; floating-point function
call __fpmath ; floating-point math
```

The `__fpmath` function is the control function for Windows 80x87 floating-point emulation.

Parameter	Description														
Function	Specifies the floating-point function to execute. The Function parameter can be one of the following values:														
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Initializes the floating-point emulator. An application calls this function when it starts. If an error occurs, the function sets the carry flag. Otherwise, it clears the flag.</td></tr><tr><td>1</td><td>Resets the floating-point emulator. The action carried out by this function is similar to the action carried out by the <code>finit</code> instruction.</td></tr><tr><td>2</td><td>Stops the floating-point emulator. An application called this function just before it ended.</td></tr><tr><td>3</td><td>Sets the handler for the coprocessor error exception (Interrupt 16). The DS:AX registers must contain the 32-bit address of the exception handler. The emulator calls the handler whenever an unmasked floating-point exception occurs. The exception handler can carry out any action--it does not have to return.</td></tr><tr><td>10</td><td>Retrieves a count of the elements on the floating-point stack, copying the count to the AX register. The number of elements is equal to the number of floating-point values on the floating-point coprocessor (if one is present) plus any additional values stored by the emulator.</td></tr><tr><td>11</td><td>Indicates whether a floating-point coprocessor is present. This function returns 1 in the AX register if a coprocessor is present. Otherwise, it returns 0.</td></tr></tbody></table>	Value	Meaning	0	Initializes the floating-point emulator. An application calls this function when it starts. If an error occurs, the function sets the carry flag. Otherwise, it clears the flag.	1	Resets the floating-point emulator. The action carried out by this function is similar to the action carried out by the <code>finit</code> instruction.	2	Stops the floating-point emulator. An application called this function just before it ended.	3	Sets the handler for the coprocessor error exception (Interrupt 16). The DS:AX registers must contain the 32-bit address of the exception handler. The emulator calls the handler whenever an unmasked floating-point exception occurs. The exception handler can carry out any action--it does not have to return.	10	Retrieves a count of the elements on the floating-point stack, copying the count to the AX register. The number of elements is equal to the number of floating-point values on the floating-point coprocessor (if one is present) plus any additional values stored by the emulator.	11	Indicates whether a floating-point coprocessor is present. This function returns 1 in the AX register if a coprocessor is present. Otherwise, it returns 0.
Value	Meaning														
0	Initializes the floating-point emulator. An application calls this function when it starts. If an error occurs, the function sets the carry flag. Otherwise, it clears the flag.														
1	Resets the floating-point emulator. The action carried out by this function is similar to the action carried out by the <code>finit</code> instruction.														
2	Stops the floating-point emulator. An application called this function just before it ended.														
3	Sets the handler for the coprocessor error exception (Interrupt 16). The DS:AX registers must contain the 32-bit address of the exception handler. The emulator calls the handler whenever an unmasked floating-point exception occurs. The exception handler can carry out any action--it does not have to return.														
10	Retrieves a count of the elements on the floating-point stack, copying the count to the AX register. The number of elements is equal to the number of floating-point values on the floating-point coprocessor (if one is present) plus any additional values stored by the emulator.														
11	Indicates whether a floating-point coprocessor is present. This function returns 1 in the AX register if a coprocessor is present. Otherwise, it returns 0.														

### Comments

Function values 4 through 9 are not used.

### Example

The following example initializes the floating-point emulator:

```
xor bx, bx; bx = 0 to initialize floating point
call __fpmath
```



### \_\_Win87EmInfo (3.1)

```
int __Win87EmInfo(pWIS, cbWin87EmInfoStruct)
Win87EmInfoStruct far *pWIS;          /* buffer to receive information */
int cbWin87EmInfoStruct;              /
* size of buffer, in bytes
/
```

The \_\_Win87EmInfo function retrieves information about the floating-point emulator, such as whether a floating-point coprocessor is present and the code and data segment addresses of the emulator.

Parameter	Description
pWIS	Points to the <u>Win87EmInfoStruct</u> structure that is to receive the floating-point emulator information.
cbWin87EmInfoStruct	Specifies the size, in bytes, of the structure that is to receive the information.

#### Returns

This function returns zero if no errors occur. Otherwise, it returns a nonzero value.

#### See Also

Win87EmInfoStruct

### `__Win87EmRestore` (3.1)

```
int __Win87EmRestore(void far *pWin87EmSaveArea, int cbWin87EmSaveArea)
void far *pWin87EmSaveArea;          /* buffer containing state */
int cbWin87EmSaveArea;                /* size,
in bytes, of buffer
/
```

The `__Win87EmRestore` function restores the states of the floating-point coprocessor (if one is present) and the floating-point emulator to the states previously saved by the `__Win87EmSave` function.

Parameter	Description
<code>pWin87EmSaveArea</code>	Points to the <code>Win87EmSaveArea</code> structure containing the state of the floating-point coprocessor and emulator. The <code>__Win87EmSave</code> function must have been used previously to fill the structure.
<code>cbWin87EmSaveArea</code>	Specifies the size, in bytes, of the structure containing the emulator state.

#### Returns

This function returns zero if the function is successful. Otherwise, it returns a nonzero value.

#### See Also

`__Win87EmSave`, [Win87EmSaveArea](#)

### \_\_Win87EmSave (3.1)

```
int __Win87EmSave(pWin87EmSaveArea, cbWin87EmSaveArea)
void far *pWin87EmSaveArea;          /* buffer to receive state */
int cbWin87EmSaveArea;                /* size,
in bytes, of buffer
/
```

The \_\_Win87EmSave function saves the current states of the floating-point coprocessor (if one is present) and the floating-point emulator, copying the states to the buffer pointed to by pWin87EmSaveArea.

An application that calls \_\_Win87EmSave should call the \_\_Win87EmRestore function before carrying out any floating-point operations.

Parameter	Description
pWin87EmSaveArea	Points to the <u>Win87EmSaveArea</u> structure that is to receive the state of the floating-point emulator.
cbWin87EmSaveArea	Specifies the size, in bytes, of the structure to receive the emulator state.

#### Returns

This function returns zero if the function is successful. Otherwise, it returns a nonzero value.

#### Comments

An application can find out the size, in bytes, of the buffer needed to save the floating-point states by using the \_\_Win87EmInfo function to retrieve the Win87EmInfoStruct structure. The SizeSaveArea member of this structure specifies the size of the buffer.

#### See Also

\_\_Win87EmInfo, \_\_Win87EmRestore, Win87EmInfoStruct, Win87EmSaveArea

## Win87EmStruct (3.1)

```
typedef struct _Win87EmInfoStruct {
    unsigned Version;
    unsigned SizeSaveArea;
    unsigned WinDataSeg;
    unsigned WinCodeSeg;
    unsigned Have80x87;
    unsigned Unused;
} Win87EmInfoStruct;
```

The Win87EmInfoStruct structure contains information about the floating-point emulator.

Member	Description
Version	Specifies the major and minor version numbers. The high-order byte specifies the major version number, the low-order byte the minor version number.
SizeSaveArea	Specifies the size, in bytes, of the buffer needed to save the floating-point emulator state. An application uses the specified size to allocate sufficient space to save the state before calling the __Win87EmSave function.
WinDataSeg	Specifies the emulator's data segment address or selector.
WinCodeSeg	Specifies the emulator's code segment address or selector.
Have80x87	Specifies the floating-point emulator flag. If an 80287 or 80387 floating-point coprocessor is present, this member is 1. Otherwise, it is 0.
Unused	Not used.

### See Also

\_\_Win87EmInfo, \_\_Win87EmSave

### Win87EmSaveArea (3.1)

```
typedef struct _Win87EmSaveArea {
    unsigned char  Save80x87Area[SIZE_80X87_AREA];
    unsigned char  SaveEmArea[];
} Win87EmSaveArea;
```

The Win87EmSaveArea structure contains the states of the floating-point coprocessor and floating-point emulator.

Member	Description
Save80x87Area	Specifies an array of values defining the state of the floating-point coprocessor if one is present. The array has the same format as data saved by an fsave instruction and consists of SIZE_80X87_AREA (94) array elements.
SaveEmArea	Specifies an array of values defining the state of the floating-point emulator. The array has the following form:

```
Have8087      db  0      ; 1 if coprocessor is present;
;otherwise, 0
              db  ?      ; reserved
              dw  ?      ; reserved
              dw  ?      ; reserved
```

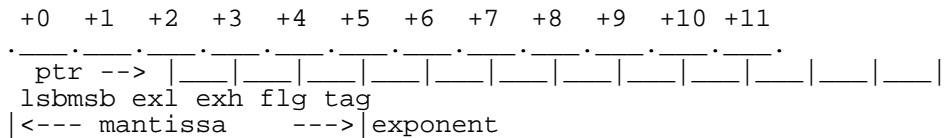
```
ControlWord  label  word ; emulator control word
CWmask      db  ?      ; exception masks
CWcntl      db  ?      ; arithmetic control flags
```

```
StatusWord   label  word ; emulator status word
SWerr       db  ?      ; exception flags
SWcc        db  ?      ; condition codes
```

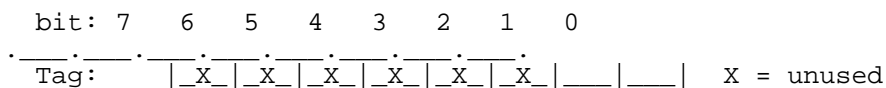
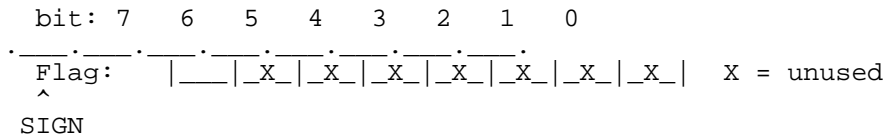
```
BASstk      dw  ?      ; offset of start of emulator
;register area
CURstk      dw  ?      ; offset of current top-of-stack
;register
LIMstk      dw  ?      ; offset of end of emulator register
;area
              dw  ? dup(?) ; reserved
```

#### Comments

The BASstk, CURstk, and LIMstk fields specify the offsets from the start of the SaveEmArea member into the emulator's register area. If BASstk and CURstk have the same value, the stack is empty. Each of the emulator's registers is 12 bytes long and has the form shown in the following illustration.



The mantissa contains the leading 1 before the decimal point in the high-order bit of the most significant byte (msb). The exponent is not biased, that is, it is a signed integer. The following illustration shows the flag and tag bytes.



^     ^  
|     |  
Special (Set for NAN or Inf) ----+ |  
ZROorINF (Set for 0 or Inf) -----+

Special    equ2  
ZROorINF  equ1  
Reg87Len  equ12

See Also

\_\_Win87EmRestore, \_\_Win87EmSave

### Floating Point Functions (3.1)

<code>__fpmath</code>	Floating Point Math
<code>__Win87EmInfo</code>	Get Floating Point Emulator Information
<code>__Win87EmRestore</code>	Restore Floating Point Emulator State
<code>__Win87EmSave</code>	Save Floating Point Emulator State

### Floating Point Structures (3.1)

Win87EmInfoStruct      Floating Point Emulator Information  
Win87EmSaveArea      Floating Point Emulator Save Area



externFP InitApp

```
push    hInstance ; instance handle  
call    InitApp
```

```
or ax,ax ; zero if error  
jz error_handler
```

The InitApp function creates the application queue and installs application-support routines, such as the signal procedure, version-specific resource loaders, and the divide-by-zero interrupt routine.

Parameter	Description
hInstance	Identifies the task to be initialized. This parameter must have been previously supplied by Windows.

#### Returns

This function returns a nonzero value in the AX register if successful. Otherwise, it returns zero in the AX register to indicate an error.

#### See Also

[InitTask](#)

```
externFP InitTask
```

```
call    InitTask    ; Initialize a task.
```

The `InitTask` function initializes the task by setting registers, setting up the command line, and initializing the heap. This must be the first function called by the startup routine for the application.

#### Returns

This function returns 1 in the AX register and fills the CX, DX, ES:BX, SI, and DI registers with information about the new task, if the function is successful. Otherwise, it returns zero in the AX register to indicate an error.

#### Comments

When the function is successful, other registers contain the following values:

Register	Value
CX	Contains the stack limit, in bytes. The startup routines should check the limit to ensure there is a minimum of 100 bytes in the stack.
DI	Contains the instance handle for the new task. The startup routine passes this address to the <u>WinMain</u> function.
DX	Contains an <code>nCmdShow</code> parameter. The startup routine passes this parameter to the <u>WinMain</u> function for use with the <u>CreateWindow</u> function.
ES	Contains the segment address of the program segment prefix (PSP) for the new task.
ES:BX	Contains the 32-bit address of the command line (MS-DOS format). The startup routine passes this address to the <u>WinMain</u> function.
SI	Contains the instance handle for the previous instance of the application, if any. The startup routine passes this address to the <u>WinMain</u> function.

The `InitTask` function also copies the top, minimum, and bottom address offsets of the stack to the 16 bytes of reserved memory at the beginning of the automatic data segment for the application. The reserved memory has the following format:

```
    DW    0
globalW  oOldSP,0
globalW  hOldSS,5
globalW  pLocalHeap,0
globalW  pAtomTable,0
globalW  pStackTop,0
globalW  pStackMin,0
globalW  pStackBot,0
```

#### See Also

InitApp

externFP WaitEvent

```
push    taskID ; task identifier
call    WaitEvent
```

```
or ax,ax
jnzresched; nonzero if rescheduled
```

The WaitEvent function checks for a posted event and, if one is found, clears the event and returns control to the application. If no event is found, the function suspends execution of the application by calling the Windows scheduler.

Parameter	Description
taskID	Identifies the task to check events for. If this parameter is zero, the function checks events for the current task.

#### Returns

This function returns a nonzero value if the Windows scheduler has scheduled another application. Otherwise, it returns zero.

## Application Startup Functions

InitApp      Initializes a Windows application  
InitTask     Initializes a Task  
WaitEvent    Suspends Execution Until an Event

## GetWinMem32Version (3.0)

```
#include <winmem32.h>
```

```
WORD GetWinMem32Version(void)
```

The `GetWinMem32Version` function retrieves the application programming interface (API) version implemented by the `WINMEM32.DLL` dynamic-link library. This is not the version number of the library itself.

### Returns

The return value specifies the version of the 32-bit memory API implemented by `WINMEM32.DLL`. The high-order 8 bits contain the major version number, and the low-order 8 bits contain the minor version number.

## Global16PointerAlloc (3.0)

```
#include <winmem32.h>
```

```
WORD Global16PointerAlloc(wSelector, dwOffset, lpBuffer, dwSize, wFlags)
```

```
WORD wSelector; /* selector of object */
```

```
DWORD dwOffset;
```

```
byte for alias
```

```
/
```

```
LPDWORD lpBuffer;
```

```
* address of location for alias
```

```
/
```

```
DWORD dwSize;
```

```
* size of region
```

```
/
```

```
WORD wFlags;
```

```
* reserved, must be zero
```

```
/
```

The Global16PointerAlloc function converts a 16:32 pointer into a 16:16 pointer alias that the application can pass to a Windows function or to other 16:16 functions.

Parameter	Description
wSelector	Specifies the selector of the object for which an alias is to be created. This must be the selector returned by a previous call to the <a href="#">Global32Alloc</a> function.
dwOffset	Specifies the offset of the first byte for which an alias is to be created. The offset is from the first byte of the object specified by the wSelector parameter. Note that wSelector:dwOffset forms a 16:32 address of the first byte of the region for which an alias is to be created.
lpBuffer	Points to a four-byte location in memory that receives the 16:16 pointer alias for the specified region.
dwSize	Specifies the addressable size, in bytes, of the region for which an alias is to be created. This value must be no larger than 64K.
wFlags	Reserved; must be zero.

### Returns

The return value is zero if the function is successful. Otherwise, it is an error value, which can be one of the following:

- WM32\_Insufficient\_Mem
- WM32\_Insufficient\_Sels
- WM32\_Invalid\_Arg
- WM32\_Invalid\_Flags
- WM32\_Invalid\_Func

### Comments

When this function returns successfully, the location pointed to by the lpBuffer parameter contains a 16:16 pointer to the first byte of the region. This is the same byte to which wSelector:dwOffset points.

The returned selector identifies a descriptor for a data segment that has the following attributes: read-write, expand up, and small (B bit clear). The descriptor privilege level (DPL) and the granularity (the G bit) are set at the system's discretion, so you should make no assumptions regarding their settings. The DPL and requestor privilege level (RPL) are appropriate for a Windows application.

Note: An application must not change the setting of any bits in the DPL or the RPL selector. Doing so can result in a system crash and will prevent the application from running on compatible platforms.

Because of tiling schemes implemented by some systems, the offset portion of the returned 16:16 pointer is not necessarily zero.

When writing your application, you should not assume the size limit of the returned selector. Instead, assume that at least dwSize bytes can be addressed starting at the 16:16 pointer created by this function.

### See Also

[Global16PointerFree](#)

### Global16PointerFree (3.0)

```
#include <winmem32.h>
```

```
WORD Global16PointerFree(wSelector, dwAlias, wFlags)
```

```
WORD wSelector; /* selector of object */
```

```
DWORD dwAlias; /* pointer alias to free */
```

```
/
```

```
WORD wFlags;
```

```
* reserved, must be zero
```

```
/
```

The Global16PointerFree function frees the 16:16 pointer alias previously created by a call to the Global16PointerAlloc function.

Parameter	Description
wSelector	Specifies the selector of the object for which the alias is to be freed. This must be the selector returned by a previous call to the <u>Global32Alloc</u> function.
dwAlias	Specifies the 16:16 pointer alias to be freed. This must be the alias (including the original offset) returned by a previous call to the <u>Global16PointerAlloc</u> function.
wFlags	Reserved; must be zero.

#### Returns

The return value is zero if the function is successful. Otherwise, it is an error value, which can be one of the following:

- WM32\_Insufficient\_Mem
- WM32\_Insufficient\_Sels
- WM32\_Invalid\_Arg
- WM32\_Invalid\_Flags
- WM32\_Invalid\_Func

#### Comments

An application should free a 16:16 pointer alias as soon as it is no longer needed. Freeing the alias releases space in the descriptor table, a limited system resource.

#### See Also

Global16PointerAlloc

## Global32Alloc (3.0)

```
#include <winmem32.h>
```

```
WORD Global32Alloc(dwSize, lpSelector, dwMaxSize, wFlags)
DWORD dwSize;          /* size of block to allocate */
LPWORD lpSelector;    /* address of location for selector */
/* maximum size of object */
WORD wFlags;          /* sharing flag */
```

The Global32Alloc function allocates a memory object to be used as a 16:32 (USE32) code or data segment and retrieves the selector portion of the 16:32 address of the memory object. The first byte of the object is at offset 0 from this selector.

Parameter	Description
dwSize	Specifies the initial size, in bytes, of the object to be allocated. This value must be in the range 1 through (16 megabytes - 64K).
lpSelector	Points to a 2-byte location in memory that receives the selector portion of the 16:32 address of the allocated object.
dwMaxSize	Specifies the maximum size, in bytes, that the object will reach when it is reallocated by the Global32Realloc function. This value must be in the range 1 through (16 megabytes - 64 K). If the application will never reallocate this memory object, the dwMaxSize parameter should be set to the same value as the dwSize parameter.
wFlags	Depends on the return value of the <u>GetWinMem32Version</u> function. If the return value is less than 0x0101, this parameter must be zero. If the return value is greater than or equal to 0x0101, this parameter can be set to <u>GMEM_DDESHARE</u> (to make the object sharable). Otherwise, this parameter should be zero. For more information about <u>GMEM_DDESHARE</u> , see the description of the <u>GlobalAlloc</u> function.

### Returns

The return value is zero if the function is successful. Otherwise, it is an error value, which can be one of the following:

```
WM32_Insufficient_Mem
WM32_Insufficient_Sels
WM32_Invalid_Arg
WM32_Invalid_Flags
WM32_Invalid_Func
```

### Comments

If the Global32Alloc function fails, the value to which the lpSelector parameter points is zero. If the function succeeds, lpSelector points to the selector of the object. The valid range of offsets for the object referenced by this selector is 0 through (but not including) dwSize.

In Windows 3.0 and later, the largest object that can be allocated is 0x00FF0000 (16 megabytes - 64K). This is the limitation placed on WINMEM32.DLL by the current Windows kernel.

The returned selector identifies a descriptor for a data segment that has the following attributes: read-write, expand-up, and big (B bit set). The descriptor privilege level (DPL) and the granularity (the G bit) are set at the system's discretion, so you should make no assumptions regarding these settings. Because the system sets the granularity, the size of the object (and the selector size limit) may be greater than the requested size by up to 4095 bytes (4K minus 1). The DPL and requestor privilege level (RPL) will be appropriate for a Windows application.

**Note:** An application must not change the setting of any bits in the DPL or the RPL selector. Doing so can result in a system crash and will prevent the application from running on compatible platforms.

The allocated object is neither movable nor discardable but can be paged. An application should not page-lock a 32-bit memory object. Page-locking an object is useful only if the object contains code or data that is used at interrupt time, and 32-bit memory cannot be used at interrupt time.



See Also  
Global32Free, Global32Realloc

## Global32CodeAlias (3.0)

```
#include <winmem32.h>
```

```
WORD Global32CodeAlias(wSelector, lpAlias, wFlags)
WORD wSelector;          /* selector of object for alias */
LPWORD lpAlias;          /* address of location for alias selector */
/* reserved, must be zero */
```

The `Global32CodeAlias` function creates a 16:32 (USE32) code-segment alias selector for a 32-bit memory object previously created by the [Global32Alloc](#) function. This allows the application to execute code contained in the memory object.

Parameter	Description
wSelector	Specifies the selector of the object for which an alias is to be created. This must be the selector returned by a previous call to the <a href="#">Global32Alloc</a> function.
lpAlias	Points to a 2-byte location in memory that receives the selector portion of the 16:32 code-segment alias for the specified object.
wFlags	Reserved; must be zero.

### Returns

The return value is zero if the function is successful. Otherwise, it is an error value, which can be one of the following:

- WM32\_Insufficient\_Mem
- WM32\_Insufficient\_Sels
- WM32\_Invalid\_Arg
- WM32\_Invalid\_Flags
- WM32\_Invalid\_Func

### Comments

If the function fails, the value pointed to by the `lpAlias` parameter is zero. If the function is successful, `lpAlias` points to a USE32 code-segment alias for the object specified by the `wSelector` parameter. The first byte of the object is at offset 0 from the selector returned in `lpAlias`. Valid offsets are determined by the size of the object as set by the most recent call to the [Global32Alloc](#) or [Global32Realloc](#) function.

The returned selector identifies a descriptor for a code segment that has the following attributes: read-execute, nonconforming, and USE32 (D bit set). The descriptor privilege level (DPL) and the granularity (the G bit) are set at the system's discretion, so you should make no assumptions regarding their settings. The granularity will be consistent with the current data selector for the object. The DPL and requestor privilege level (RPL) are appropriate for a Windows application.

**Note:** An application must not change the setting of any bits in the DPL or the RPL selector. Doing so can result in a system crash and will prevent the application from running on compatible platforms.

An application should not call this function more than once for an object. Depending on the system, the function might fail if an application calls it a second time for a given object without first calling the [Global32CodeAliasFree](#) function for the object.

### See Also

[Global32Alloc](#), [Global32CodeAliasFree](#)

### Global32CodeAliasFree (3.0)

```
#include <winmem32.h>
```

```
WORD Global32CodeAliasFree(wSelector, wAlias, wFlags)
```

```
WORD wSelector; /* selector of object */
```

```
WORD wAlias; /* code-segment
```

```
alias selector to free
```

```
/
```

```
WORD wFlags;
```

```
* reserved, must be zero
```

```
/
```

The Global32CodeAliasFree function frees the 16:32 (USE32) code-segment alias selector previously created by a call to the Global32CodeAlias function.

Parameter	Description
wSelector	Specifies the selector of the object for which the alias is to be freed. This must be the selector returned by a previous call to the <u>Global32Alloc</u> function.
wAlias	Specifies the USE32 code-segment alias selector to be freed. This must be the alias returned by a previous call to the <u>Global32CodeAlias</u> function.
wFlags	Reserved; must be zero.

#### Returns

The return value is zero if the function is successful. Otherwise, it is an error value, which can be one of the following:

WM32\_Insufficient\_Mem

WM32\_Insufficient\_Sels

WM32\_Invalid\_Arg

WM32\_Invalid\_Flags

WM32\_Invalid\_Func

#### See Also

Global32CodeAlias

## Global32Free (3.0)

```
#include <winmem32.h>
```

```
WORD Global32Free(wSelector, wFlags)
```

```
WORD wSelector; /* selector of object to free */
```

```
WORD wFlags; /* reserved, must be
```

```
zero
```

```
/
```

```
*
```

The `Global32Free` function frees an object previously allocated by the [Global32Alloc](#) function.

Parameter	Description
wSelector	Specifies the selector of the object to be freed. This must be the selector returned by a previous call to the <a href="#">Global32Alloc</a> function.
wFlags	Reserved; must be zero.

### Returns

The return value is zero if the function is successful. Otherwise, it is an error value, which can be one of the following:

- WM32\_Insufficient\_Mem
- WM32\_Insufficient\_Sels
- WM32\_Invalid\_Arg
- WM32\_Invalid\_Flags
- WM32\_Invalid\_Func

### Comments

The [Global32Alloc](#) function frees the object itself; it also frees all aliases created for the object by the 32-bit memory application programming interface (API).

Note: Before terminating, an application must call this function to free each object allocated by the [Global32Alloc](#) function to ensure that all aliases created for the object are freed.

### See Also

[Global32Alloc](#), [Global32Realloc](#)

## Global32Realloc (3.0)

```
#include <winmem32.h>
```

```
WORD Global32Realloc(wSelector, cNew, flags)
```

```
WORD wSelector; /* selector of object to reallocate */
```

```
DWORD cNew; /* new size of object */
```

```
WORD flags;
```

```
* reserved, must be zero
```

```
/
```

The Global32Realloc function changes the size of a 32-bit memory object previously allocated by the Global32Alloc function.

Parameter	Description
wSelector	Specifies the selector of the object to be changed. This must be the selector returned by a previous call to the <u>Global32Alloc</u> function.
cNew	Specifies the new size, in bytes, of the object. This value must be greater than zero and less than or equal to the size specified by the dwMaxSize parameter of the <u>Global32Alloc</u> function call that created the object.
flags	Reserved; must be zero.

### Returns

The return value is zero if the function is successful. Otherwise, it is an error value, which can be one of the following:

- WM32\_Insufficient\_Mem
- WM32\_Insufficient\_Sels
- WM32\_Invalid\_Arg
- WM32\_Invalid\_Flags
- WM32\_Invalid\_Func

### Comments

If this function fails, the previous state of the object is unchanged. If the function succeeds, it updates the state of the object and the state of all aliases to the object created by the 32-bit memory application programming interface (API) functions. For this reason, an application must call the Global32Realloc function to change the size of the object. Using other Windows functions to manipulate the object results in corrupted aliases.

This function does not change the selector specified by the wSelector parameter. If this function succeeds, the new valid range of offsets for the selector is zero through (but not including) cNew.

The system determines the appropriate granularity of the object. As a result, the size of the object (and the selector size limit) may be greater than the requested size by up to 4095 bytes (4K minus 1).

### See Also

Global32Alloc, Global32Free

## ABORTDOC

short Escape(hdc, ABORTDOC, NULL, NULL, NULL)

The ABORTDOC printer escape is maintained for backwards compatibility. Applications written for Windows 3.1 should use the AbortDoc function.

This escape stops the current job and erases everything the application has written to the device since the last ENDDOC escape.

The ABORTDOC escape should be used to stop:

- Printing operations that do not specify an Abort function by using the SETABORTPROC escape.
- Printing operations that have not yet reached their first call to the NEWFRAME or NEXTBAND escape.

Parameter	Description
hdc	HDC Identifies the device context.

### Returns

This escape does not return a value.

### Comments

If an application encounters a printing error, it should not try to stop the operation by using the Escape function with either the ENDDOC or ABORTDOC escape. Graphics device interface (GDI) automatically terminates the operation before returning the error value.

If the application displays a dialog box to allow the user to cancel the print operation, it must send the ABORTDOC escape before destroying the dialog box.

The application must send the ABORTDOC escape before freeing the procedure-instance address of the Abort function, if any.

### See Also

Escape, AbortDoc

## BANDINFO

short Escape(hdc, BANDINFO, sizeof(BANDINFOSTRUCT), lpInData, lpOutData)

The BANDINFO printer escape is maintained for backwards compatibility. Applications written for Windows 3.1 should send both text and graphics in every band.

This escape copies information about a device with banding capabilities to a structure pointed to by the lpOutData parameter. It is implemented only for devices that use banding to send output to the printer.

Banding is the property of an output device that allows a page of output to be stored in a metafile and divided into bands, each of which is sent to the device to create a complete page.

The information copied to the structure pointed to by the lpOutData parameter includes:

- A value that indicates whether there are graphics in the next band.
- A value that indicates whether there is text on the page.
- A RECT structure that contains a bounding rectangle for all graphics on the page.

If no data is returned, the lpOutData parameter is NULL.

The lpInData parameter specifies information sent by the application to the printer driver. This information is read by the driver only on the first call to the BANDINFO escape on a page.

Parameter	Description
hdc	HDC Identifies the device context.
lpInData	BANDINFOSTRUCT FAR * Points to a BANDINFOSTRUCT structure that contains information to be passed to the driver. For more information about this structure, see the following Comments section.
lpOutData	BANDINFOSTRUCT FAR * Points to a BANDINFOSTRUCT structure that contains information returned by the driver. For more information about this structure, see the following Comments section.

### Returns

The return value specifies the outcome of the escape. This value is 1 if the escape is successful. It is zero if the function fails or is not implemented by the driver.

### Comments

The BANDINFOSTRUCT structure contains information about the contents of a page and supplies a bounding rectangle for graphics on the page. This structure has the following form:

```
typedef struct tagBANDINFOSTRUCT {
    BOOL    fGraphics;
    BOOL    fText;
    RECT    rcGraphics;
} BANDINFOSTRUCT;
```

Following are the members in the BANDINFOSTRUCT structure:

fGraphics	Specifies nonzero if graphics are or are expected to be on the page or in the band. Otherwise, it is zero.
fText	Specifies nonzero if text is or is expected to be on the page or in the band. Otherwise, it is zero.
rcGraphics	Contains a <u>RECT</u> structure that supplies a bounding region for all graphics on the page.

The meaning of these members depends on which parameter contains the structure, as follows.

Member	When Used in lpInData	When used in lpOutData
fGraphics	Nonzero if the application informs the driver that graphics are on the page	Nonzero if the driver informs the application that it expects graphics in this band
fText	Nonzero if the application informs the driver that text is on the page	Nonzero if the driver informs the application that it expects text in this band
rcGraphics	Bounding rectangle supplied for all graphics on the page	No valid return data

An application should call this escape immediately after each call to the NEXTBAND escape. The BANDINFO escape is in reference to the band that the driver returned to the NEXTBAND escape.

An application should use this escape in the following manner:

- On the first band, the driver may give the application a full-page band and ask for text only (the fGraphics member is set to zero and the fText member is set to nonzero). Then the application sends only text to the driver.
- If in the first band the application indicates that it has graphics (the fGraphics member is set to nonzero) or the driver encounters vector fonts, the driver bands the rest of the page.
- If there are no graphics or vector fonts, the next NEXTBAND escape returns an empty rectangle to indicate that the application should move on to the next page.
- If there are graphics but no vector fonts (the application sets the fGraphics member to nonzero, but there are no graphics in the first full-page text band), the driver may optionally band only into the rectangle the application passes for subsequent bands. This rectangle bounds all graphics on the page.
- If there are vector fonts, the driver bands the entire width and depth of the page with the fText member set to nonzero. It also sets the fGraphics flag to nonzero if the application has set it.

The driver assumes that an application using the BANDINFO escape only sends text in the first full-page text band because that is all the driver has requested. Therefore, if the driver encounters a vector font or graphics in the band, it assumes they were generated by a text primitive and sets the fText member to nonzero for all subsequent graphics bands, so they can be output as graphics. If the application does not meet this expectation, the image still generates properly, but the driver spends time sending spurious text primitives to graphics bands.

Older drivers written before the BANDINFO escape was designed use full-page banding for text. If a particular driver does not support the BANDINFO escape but sets the RC\_BANDING raster capability, the application can detect full-page banding for text by determining if the first band on the page covers the entire page.



## BEGIN\_PATH

short Escape(hdc, BEGIN\_PATH, NULL, NULL, NULL)

The BEGIN\_PATH printer escape opens a path. A path is a connected sequence of primitives drawn in succession to form a single polyline or polygon. Paths enable applications to draw complex borders, filled shapes, and clipping regions by supplying a collection of other primitives to define the desired shape.

Printer escapes supporting paths enable applications to render images on sophisticated devices, such as PostScript printers, without generating huge polygons to simulate the images.

To draw a path, an application first issues the BEGIN\_PATH escape. Then it draws the primitives defining the border of the desired shape and issues an END\_PATH escape, which includes a parameter specifying how the path is to be rendered.

Parameter	Description
hdc	HDC Identifies the device context.

### Returns

The return value specifies the current path nesting level. This value is the number of calls to the BEGIN\_PATH escape without a corresponding call to the END\_PATH escape if the escape is successful. Otherwise, the return value is zero.

### Comments

This escape is used only by PostScript printer drivers.

An application may begin a subpath within another path. If the subpath is closed, it is treated just like a polygon. If it is open, it is treated just like a polyline.

An application may use the CLIP\_TO\_PATH escape to define a clipping region corresponding to the interior or exterior of the currently open path.

## CLIP\_TO\_PATH

short Escape(hdc, CLIP\_TO\_PATH, sizeof(int), lpClipMode, NULL)

The CLIP\_TO\_PATH printer escape defines a clipping region bounded by the currently open path. It enables the application to save and restore the current clipping region and to set up an inclusive or exclusive clipping region bounded by the currently open path. If the path defines an inclusive clipping region, portions of primitives falling outside the interior bounded by the path are clipped. If the path defines an exclusive clipping region, portions of primitives falling inside the interior are clipped.

Parameter	Description
hdc	HDC Identifies the device context.
lpClipMode	LPINT Points to a short integer that specifies the clipping mode. It can be one of the following values:

Value	Meaning
CLIP_SAVE (0)	Saves the current clipping region.
CLIP_RESTORE (1)	Restores the previous clipping region.
CLIP_INCLUSIVE (2)	Sets an inclusive clipping region.
CLIP_EXCLUSIVE (3)	Sets an exclusive clipping region.

### Returns

The return value specifies the outcome of the escape. This value is nonzero if the escape is successful. Otherwise, it is zero.

### Comments

This escape is used only by PostScript printer drivers.

To clip a set of primitives against a path, an application should follow these steps:

- 1 Save the current clipping region by using the CLIP\_TO\_PATH escape.
- 2 Begin a path with the BEGIN\_PATH escape.
- 3 Draw the primitives bounding the clipping region.
- 4 Close the path with the END\_PATH escape.
- 5 Set the clipping region by using the CLIP\_TO\_PATH escape.
- 6 Draw the primitives to be clipped.
- 7 Restore the original clipping region by using the CLIP\_TO\_PATH escape.

## DEVICEDATA

short Escape(hdc, DEVICEDATA, nCount, lpInData, lpOutData)

The DEVICEDATA printer escape is identical to the PASSTHROUGH escape. For further information, see the description of PASSTHROUGH.

## DRAFTMODE

short Escape(hdc, DRAFTMODE, sizeof(int), lpDraftMode, NULL)

The DRAFTMODE printer escape turns draft mode off or on. Turning draft mode on instructs the driver to print faster and with lower quality, if necessary. The draft mode can be changed only at page boundaries (for example, after a NEWFRAME escape directing the driver to advance to a new page).

Parameter	Description						
hdc	HDC Identifies the device context.						
lpDraftMode	LPINT Points to a short integer that specifies the draft mode. It can be one of the following values:						
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Specifies draft mode off.</td></tr><tr><td>1</td><td>Specifies draft mode on.</td></tr></tbody></table>	Value	Meaning	0	Specifies draft mode off.	1	Specifies draft mode on.
Value	Meaning						
0	Specifies draft mode off.						
1	Specifies draft mode on.						

### Returns

The return value specifies the outcome of the escape. This value is positive if the escape is successful. Otherwise, it is zero or negative.

### Comments

The default draft mode is off.

## DRAWPATTERNRECT

short Escape(hdc, DRAWPATTERNRECT, sizeof(PRECTSTRUCT), lpInData, NULL)

The DRAWPATTERNRECT printer escape creates a pattern, gray scale, or solid black rectangle by using the pattern and rule capabilities of Page Control Language (PCL) on Hewlett-Packard LaserJet or LaserJet-compatible printers. A gray scale is a gray pattern that contains a specific mixture of black and white pixels.

Parameter	Description
hdc	HDC Identifies the device context.
lpInData	PRECT_STRUCT FAR * Points to a PRECT_STRUCT structure that describes the rectangle. For more information on this structure, see the following Comments section.

### Returns

The return value specifies the outcome of the escape. This value is 1 if the escape is successful. Otherwise, it is zero.

### Comments

The lpInData parameter points to a PRECT\_STRUCT structure that defines the rectangle to be created. This structure has the following form:

```
struct PRECT_STRUCT {
    POINT ptPosition;
    POINT ptSize;
    WORD wStyle;
    WORD wPattern;
};
```

Following are the members in the PRECT\_STRUCT structure:

ptPosition	Specifies the upper-left corner of the rectangle.
ptSize	Specifies the lower-right corner of the rectangle.
wStyle	Specifies the type of pattern. It can be one of the following values:
Value	Meaning
0	Black rule
1	White rule that erases bitmap data previously written to same area (available on the HP LaserJet IIP only)
2	Gray scale
3	HP-defined
wPattern	Specifies the pattern. It is ignored for a black rule. It specifies the percentage of gray for a gray-scale pattern. It represents one of six patterns defined by Hewlett-Packard.

### Comments

The output of the DRAWPATTERNRECT escape does not go through the graphics banding bitmap; it is sent to the printer in the text band. An application can use this escape to print line and block graphics without using graphics banding at all. Because many applications use only horizontal and vertical lines or blocks in graphic output, this is a significant optimization.

An application should use the QUERYESCSUPPORT escape to determine whether a device is capable of drawing patterns and rules before using the DRAWPATTERNRECT escape. If an application uses the BANDINFO escape, all patterns and rectangles sent by using DRAWPATTERNRECT should be treated as text and sent on a text band.

Applications that use the DRAWPATTERNRECT escape must observe two limitations. First, rules drawn with DRAWPATTERNRECT are not subject to clipping regions in the device context. Second, applications should not try to erase patterns and rules created with DRAWPATTERNRECT by placing opaque objects over them. If the printer supports white rules, these can be used to erase patterns created by DRAWPATTERNRECT. If the printer does not support white rules, there is no method for erasing these patterns.

If an application cannot use the DRAWPATTERNRECT escape, it should generally use the PatBlt function instead. (If PatBlt is used to print a black rectangle, the application should use the BLACKNESS raster operator.) If the device is a plotter, the application should use the Rectangle function.

## ENABLEDUPLEX

short Escape(hdc, ENABLEDUPLEX, sizeof(WORD), lpInData, NULL)

The ENABLEDUPLEX printer escape is maintained for backwards compatibility. Applications written for Windows 3.1 should use the `ExtDeviceMode` function. An application can determine whether an output device is capable of creating duplex output by checking the `DM_DUPLEX` bit of the `dmFields` member in the `DEVMODE` structure.

This escape enables the duplex printing capabilities of a printer. A device that possesses duplex printing capabilities is able to print on both sides of the output medium.

Parameter	Description								
<code>hdc</code>	HDC Identifies the device context.								
<code>lpInData</code>	<code>LPWORD</code> Points to an unsigned 16-bit integer that specifies whether duplex or simplex printing is used. It can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Simplex</td></tr><tr><td>1</td><td>Duplex with vertical binding</td></tr><tr><td>2</td><td>Duplex with horizontal binding</td></tr></tbody></table>	Value	Meaning	0	Simplex	1	Duplex with vertical binding	2	Duplex with horizontal binding
Value	Meaning								
0	Simplex								
1	Duplex with vertical binding								
2	Duplex with horizontal binding								

### Returns

The return value specifies the outcome of the escape. This value is 1 if the escape is successful. Otherwise, it is zero.

### Comments

An application should use the `QUERYESCSUPPORT` escape to determine whether an output device is capable of creating duplex output. If `QUERYESCSUPPORT` returns a nonzero value, the application should send the `ENABLEDUPLEX` escape even if simplex printing is desired. This procedure guarantees replacement of any values set in the driver-specific dialog box. If duplex printing is enabled and an uneven number of `NEXTFRAME` escapes are sent to the driver prior to the `ENDDOC` escape, the driver ejects an additional page before ending the print job.

## ENABLEPAIRKERNING

short Escape(hdc, ENABLEPAIRKERNING, sizeof(int), lpNewKernFlag, lpOldKernFlag)

The ENABLEPAIRKERNING printer escape enables or disables the ability of the driver to kern character pairs automatically. Kerning is the process of adding or subtracting space between characters in a string of text.

When pair kerning is enabled, the driver automatically kernes those pairs of characters that are listed in the character-pair kerning table for the font. The driver reflects this kerning both on the printer and in the GetTextExtent function calls.

Parameter	Description
hdc	HDC Identifies the device context.
lpNewKernFlag	<u>LPINT</u> Points to a short-integer value that specifies whether automatic pair kerning is to be enabled (1) or disabled (zero).
lpOldKernFlag	<u>LPINT</u> Points to a short-integer value that receives the previous automatic pair-kerning value.

### Returns

The return value specifies the outcome of the escape. This value is 1 if the escape is successful. It is zero if the escape is not successful or not implemented.

### Comments

The default state of this escape is zero; automatic character-pair kerning is disabled.

A driver does not have to support the ENABLEPAIRKERNING escape just because it supplies the character-pair kerning table to the application by using the GETPAIRKERNTABLE escape. When the GETPAIRKERNTABLE escape is supported but the ENABLEPAIRKERNING escape is not, the application must properly space the kerned characters on the output device by using the ExtTextOut function.

## ENABLERELATIVEWIDTHS

short Escape(hdc, ENABLERELATIVEWIDTHS, sizeof(int), lpNewWidthFlag, lpOldWidthFlag)

The ENABLERELATIVEWIDTHS printer escape enables or disables relative character widths. When relative widths are disabled (the default), the width of each character can be expressed as a number of device units. This method guarantees that the extent of a string will equal the sum of the extents of the characters in the string. This allows applications to build an extent table by using one-character GetTextExtent function calls.

When relative widths are enabled, the sum of a string may not equal the sum of the widths of the characters. Applications that enable this feature are expected to retrieve the extent table for the font and compute relatively scaled string widths.

Parameter	Description
hdc	HDC Identifies the device context.
lpNewWidthFlag	<u>LPINT</u> Points to a short integer that specifies whether relative widths are to be enabled (1) or disabled (zero).
lpOldWidthFlag	<u>LPINT</u> Points to a short integer that receives the previous relative character width value.

### Returns

The return value specifies the outcome of the escape. This value is 1 if the escape is successful. It is zero if the escape is not successful or not implemented.

### Comments

The default state of this escape is zero; relative character widths are disabled.

When the ENABLERELATIVEWIDTHS escape is enabled, the values specified as font units and accepted and returned by the escapes described in this topic are returned in the relative units of the font.

It is assumed that only linear-scaling devices are dealt with in a relative mode. Nonlinear-scaling devices do not implement this escape.



## ENDDOC

short Escape(hdc, ENDDOC, NULL, NULL, NULL)

The ENDDOC printer escape is maintained for backwards compatibility. Applications written for Windows 3.1 should use the EndDoc function.

This escape ends a print job started by a STARTDOC escape.

Parameter	Description
hdc	HDC Identifies the device context.

### Returns

The return value specifies the outcome of the escape. This value is positive if the escape is successful. Otherwise, it is zero or negative.

### Comments

The ENDDOC escape should not be used inside metafiles.

## END\_PATH

short Escape(hdc, END\_PATH, sizeof(PATH\_INFO), lpInData, NULL)

The END\_PATH printer escape ends a path. A path is a connected sequence of primitives drawn in succession to form a single polyline or polygon. Paths enable applications to draw complex borders, filled shapes, and clipping regions by supplying a collection of other primitives to define the desired shape.

Printer escapes that support paths enable applications to render images on sophisticated devices, such as PostScript printers, without generating huge polygons to simulate them.

To draw a path, an application first issues the BEGIN\_PATH escape. Then it draws the primitives defining the border of the desired shape and issues an END\_PATH escape.

The END\_PATH escape takes, as a parameter, a pointer to a structure specifying the manner in which the path is to be rendered. The structure specifies whether or not the path is to be drawn and whether it is open or closed. Open paths define polylines, and closed paths define fillable polygons.

Parameter	Description
hdc	HDC Identifies the device context.
lpInData	PATH_INFO FAR * Points to a PATH_INFO structure that defines how the path is to be rendered. For more information about this structure, see the following Comments section.

### Returns

The return value specifies the current path nesting level. This value is the number of BEGIN\_PATH escape calls without a corresponding END\_PATH call if the escape is successful. Otherwise, it is -1.

### Comments

This escape is used only by PostScript printer drivers.

An application may begin a subpath within another path. If the subpath is closed, it is treated just like a polygon. If it is open, it is treated just like a polyline.

An application may use the CLIP\_TO\_PATH escape to define a clipping region corresponding to the interior or exterior of the currently open path.

The lpInData parameter points to a PATH\_INFO structure that specifies how to render the path. This structure has the following form:

```
struct PATH_INFO {
    short    RenderMode;
    BYTEFillMode;
    BYTEBkMode;
    LOGPEN   Pen;
    LOGBRUSH Brush;
    DWORD    BkColor;
};
```

Following are the members in the PATH\_INFO structure:

RenderMode	Specifies how the path is to be rendered. It can be one of the following values:
Value	Meaning
NO_DISPLAY (0)	Path is not drawn.
OPEN (1)	Path is drawn as an open polygon.
CLOSED (2)	Path is drawn as a closed polygon.
FillMode	Specifies how the path is to be filled. It can be one of the following values:
Value	Meaning
ALTERNATE (1)	Fill is done using the alternate fill algorithm.
WINDING (2)	Fill is done using the winding fill algorithm.
BkMode	Specifies the background mode for filling the path. It can be one of the following values:
Value	Meaning
OPAQUE	Background is filled with the background color before the brush is drawn.
TRANSPARENT	Background is not changed.

Pen	Specifies the pen with which the path is to be drawn. If the RenderMode function is set to the NO_DISPLAY value, the pen is ignored.
Brush	Specifies the brush with which the path is to be filled. If the RenderMode function is set to the NO_DISPLAY or OPEN value, the brush is ignored.
BkColor	Specifies the color with which the path is filled if the BkMode function is set to the <u>OPAQUE</u> value.

## ENUMPAPERBINS

short Escape(hdc, ENUMPAPERBINS, sizeof(int), lpNumBins, lpOutData)

The ENUMPAPERBINS printer escape is maintained for backwards compatibility. Applications written for Windows 3.1 should call the DeviceCapabilities function with the DC\_BINNAMES index to retrieve the number of available paper bins and the name of each bin.

This escape retrieves attribute information about a specified number of paper bins. The GETSETPAPERBINS escape retrieves the number of bins available on a printer.

Parameter	Description
hdc	HDC Identifies the device context.
lpNumBins	<u>LPINT</u> Points to an integer that specifies the number of bins for which information is to be retrieved.
lpOutData	<u>LPSTR</u> Points to a structure to which information about the paper bins is copied. The size of the structure depends on the number of bins for which information was requested. For a description of this structure, see the following Comments section.

### Returns

The return value specifies the outcome of the escape. This value is 1 if the escape is successful. It is zero if the escape is not successful or not implemented.

### Comments

The structure to which the lpOutData parameter points consists of two arrays. The first is an array of short integers containing the paper-bin identifier numbers in the following form:

```
short BinList[cBinMax]
```

The number of integers in the array (the cBinMax value) is equal to the value pointed to by the lpNumBins parameter.

The second array in the structure to which lpOutData points is an array of characters in the following form:

```
char PaperNames[cBinMax][cchBinName]
```

The cBinMax value is equal to the value pointed to by the lpNumBins parameter. The cchBinName value is the length of each string (currently 24).

## ENUMPAPERMETRICS

short Escape(hdc, ENUMPAPERMETRICS, sizeof(int), lpMode, lpOutData)

The ENUMPAPERMETRICS printer escape performs one of two functions according to the mode:

- It determines the number of paper types supported and returns this value, which can then be used to allocate an array of RECT structures.
- It returns one or more RECT structures that define the areas on the page that can receive an image.

This escape is provided only for backward compatibility. An application should call the DeviceCapabilities function with the DC\_PAPERSIZE index to discover the number of available paper sizes and the dimensions of each size.

Parameter	Description						
hdc	HDC Identifies the device context.						
lpMode	<u>LPINT</u> Points to an integer that specifies the mode for the escape. It can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Return value indicates how many <u>RECT</u> structures are required to contain the information about the available paper types.</td></tr><tr><td>1</td><td>Array of <u>RECT</u> structures to which the lpOutData parameter points is filled with the information.</td></tr></tbody></table>	Value	Meaning	0	Return value indicates how many <u>RECT</u> structures are required to contain the information about the available paper types.	1	Array of <u>RECT</u> structures to which the lpOutData parameter points is filled with the information.
Value	Meaning						
0	Return value indicates how many <u>RECT</u> structures are required to contain the information about the available paper types.						
1	Array of <u>RECT</u> structures to which the lpOutData parameter points is filled with the information.						
lpOutData	<u>LPRECT</u> Points to an array of <u>RECT</u> structures that return all the areas capable of receiving an image.						

### Returns

The return value is positive if the escape is successful. The value is zero if the escape is not implemented and negative if an error occurred.

## EPSPRINTING

short Escape(hdc, EPSPRINTING, sizeof(BOOL), lpBool, NULL)

The EPSPRINTING printer escape suppresses the output of the Windows PostScript header control section, which is about 7K. If an application uses this escape, no graphics device interface (GDI) calls are allowed.

Parameter	Description
hdc	HDC Identifies the device context.
lpBool	BOOL FAR * Points to a Boolean value that indicates whether downloading should be enabled (nonzero) or disabled (zero).

### Returns

The return value is positive if the escape is successful. This value is zero if the escape is not implemented and negative if an error occurred.

### Comments

This escape is used only by PostScript printer drivers.

## EXT\_DEVICE\_CAPS

short Escape(hdc, EXT\_DEVICE\_CAPS, sizeof(int), lpIndex, lpCaps)

The EXT\_DEVICE\_CAPS printer escape retrieves information about device-specific capabilities. It supplements the GetDeviceCaps function.

Parameter	Description														
hdc	HDC Identifies the device context.														
lpIndex	<u>LPINT</u> Points to a short integer that specifies the index of the capability to be retrieved. It can be any one of the following values:														
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>R2_CAPS (1)</td><td>The lpCaps parameter indicates which of the 16 binary raster operations the device driver supports. A bit will be set for each supported raster operation. For further information, see the description of the <u>SetROP2</u> function.</td></tr><tr><td>PATTERN_CAPS (2)</td><td>The lpCaps parameter returns the maximum dimensions of a pattern brush bitmap. The low-order word of the capability value contains the maximum width of a pattern brush bitmap, and the high-order word contains the maximum height.</td></tr><tr><td>PATH_CAPS (3)</td><td>The lpCaps parameter indicates whether the device is capable of creating paths by using alternate and winding interiors, and whether the device can do exclusive or inclusive clipping to path interiors. The path capabilities are obtained by using the logical OR operation on the following values:  PATH_ALTERNATE (1) PATH_WINDING (2) PATH_INCLUSIVE (4) PATH_EXCLUSIVE (8)</td></tr><tr><td>POLYGON_CAPS (4)</td><td>The lpCaps parameter returns the maximum number of polygon points supported by the device. The capability value is an unsigned value specifying the maximum number of points.</td></tr><tr><td>PATTERN_COLOR_CAPS (5)</td><td>The lpCaps parameter indicates whether the device can convert monochrome pattern bitmaps to color. The capability value is 1 if the device can do pattern bitmap color conversions and zero if it cannot.</td></tr><tr><td>R2_TEXT_CAPS (6)</td><td>The lpCaps parameter indicates whether the device is capable of performing binary raster operations on text. The low-order word of the capability value specifies which raster operations are supported for text. A bit is set for each supported raster operation, as in the R2_CAPS escape. The high-order word specifies the type of text to which the raster capabilities apply. It is obtained by applying the logical OR operation to the following values together:  RASTER_TEXT (1) DEVICE_TEXT (2) VECTOR_TEXT (4)</td></tr></tbody></table>	Value	Meaning	R2_CAPS (1)	The lpCaps parameter indicates which of the 16 binary raster operations the device driver supports. A bit will be set for each supported raster operation. For further information, see the description of the <u>SetROP2</u> function.	PATTERN_CAPS (2)	The lpCaps parameter returns the maximum dimensions of a pattern brush bitmap. The low-order word of the capability value contains the maximum width of a pattern brush bitmap, and the high-order word contains the maximum height.	PATH_CAPS (3)	The lpCaps parameter indicates whether the device is capable of creating paths by using alternate and winding interiors, and whether the device can do exclusive or inclusive clipping to path interiors. The path capabilities are obtained by using the logical OR operation on the following values:  PATH_ALTERNATE (1) PATH_WINDING (2) PATH_INCLUSIVE (4) PATH_EXCLUSIVE (8)	POLYGON_CAPS (4)	The lpCaps parameter returns the maximum number of polygon points supported by the device. The capability value is an unsigned value specifying the maximum number of points.	PATTERN_COLOR_CAPS (5)	The lpCaps parameter indicates whether the device can convert monochrome pattern bitmaps to color. The capability value is 1 if the device can do pattern bitmap color conversions and zero if it cannot.	R2_TEXT_CAPS (6)	The lpCaps parameter indicates whether the device is capable of performing binary raster operations on text. The low-order word of the capability value specifies which raster operations are supported for text. A bit is set for each supported raster operation, as in the R2_CAPS escape. The high-order word specifies the type of text to which the raster capabilities apply. It is obtained by applying the logical OR operation to the following values together:  RASTER_TEXT (1) DEVICE_TEXT (2) VECTOR_TEXT (4)
Value	Meaning														
R2_CAPS (1)	The lpCaps parameter indicates which of the 16 binary raster operations the device driver supports. A bit will be set for each supported raster operation. For further information, see the description of the <u>SetROP2</u> function.														
PATTERN_CAPS (2)	The lpCaps parameter returns the maximum dimensions of a pattern brush bitmap. The low-order word of the capability value contains the maximum width of a pattern brush bitmap, and the high-order word contains the maximum height.														
PATH_CAPS (3)	The lpCaps parameter indicates whether the device is capable of creating paths by using alternate and winding interiors, and whether the device can do exclusive or inclusive clipping to path interiors. The path capabilities are obtained by using the logical OR operation on the following values:  PATH_ALTERNATE (1) PATH_WINDING (2) PATH_INCLUSIVE (4) PATH_EXCLUSIVE (8)														
POLYGON_CAPS (4)	The lpCaps parameter returns the maximum number of polygon points supported by the device. The capability value is an unsigned value specifying the maximum number of points.														
PATTERN_COLOR_CAPS (5)	The lpCaps parameter indicates whether the device can convert monochrome pattern bitmaps to color. The capability value is 1 if the device can do pattern bitmap color conversions and zero if it cannot.														
R2_TEXT_CAPS (6)	The lpCaps parameter indicates whether the device is capable of performing binary raster operations on text. The low-order word of the capability value specifies which raster operations are supported for text. A bit is set for each supported raster operation, as in the R2_CAPS escape. The high-order word specifies the type of text to which the raster capabilities apply. It is obtained by applying the logical OR operation to the following values together:  RASTER_TEXT (1) DEVICE_TEXT (2) VECTOR_TEXT (4)														

POLYMODE\_CAPS (7)

The lpcaps parameter indicates which poly modes are supported by the printer driver. The capability value is obtained by using the bitwise OR operator to combine a bit in the corresponding position for each supported poly mode. For example, if the printer supports the PM\_POLYSCANLINE and PM\_BEZIER poly modes, the capability value would be:  
(1 << PM\_POLYSCANLINE) | (PM\_BEZIER)

lpCaps            LPDWORD Points to a 32-bit integer to which the capabilities will be copied.

Returns

The return value is nonzero if the specified extended capability is supported. This value is zero if the capability is not supported.

Comments

This escape is used only by PostScript printer drivers.



## EXTTEXTOUT

short Escape(hdc, EXTTEXTOUT, sizeof(EXTTEXT\_STRUCT), lpInData, NULL)

The EXTTEXTOUT printer escape provides an efficient way for an application to call the graphics device interface (GDI) TextOut function when justification, letter spacing, or kerning is involved.

This function is provided only for backward compatibility. New applications should use the GDI ExtTextOut function instead.

Parameter	Description
hdc	HDC Identifies the device context.
lpInData	EXTTEXT_STRUCT FAR * Points to an EXTTEXT_STRUCT structure that specifies the initial position, characters, and character widths of the string. For more information about this structure, see the following Comments section.

### Returns

The return value specifies the outcome of the escape. This value is 1 if the escape is successful. It is zero if the escape is not successful or not implemented.

### Comments

The EXTTEXT\_STRUCT structure has the following form:

```
struct EXTTEXT_STRUCT {
    WORD    x;
    WORD    y;
    LPWORD  lpText;
    LPWORD  lpWidths;
};
```

Following are the members in the EXTTEXT\_STRUCT structure:

x	Specifies the x-coordinate of the upper-left corner of the string's starting point.
y	Specifies the y-coordinate of the upper-left corner of the string's starting point.
lpText	Points to an array of cch character codes, where cch is the number of bytes in the string (cch is also the number of words in the width array).
lpWidths	Points to an array of cch character widths to use when printing the string. The first character appears at (x,y), the second at (x + lpWidths[0],y), the third at (x + lpWidths[0] + lpWidths[1],y), and so on. These character widths are specified in the font units of the currently selected font. (The character widths are always equal to device units, unless the application has enabled relative character widths.) The units contained in the width array are specified as font units of the device.

## FLUSHOUTPUT

short Escape(hdc, FLUSHOUTPUT, NULL, NULL, NULL)

The FLUSHOUTPUT printer escape clears all output from the device's buffer.

Parameter	Description
hdc	HDC Identifies the device context.

### Returns

The return value specifies the outcome of the escape. This value is greater than zero if the escape is successful. Otherwise, it is less than zero.

## GETCOLORTABLE

short Escape(hdc, GETCOLORTABLE, sizeof(int), lpIndex, lpColor)

The GETCOLORTABLE printer escape retrieves an RGB color-table entry and copies it to the location specified by the lpColor parameter.

Parameter	Description
hdc	HDC Identifies the device context.
lpIndex	<u>LPINT</u> Points to a short integer that specifies the index of a color-table entry. Color-table indexes start at zero for the first table entry.
lpColor	<u>LPDWORD</u> Points to the long integer that will receive the <u>RGB</u> color value for the given entry.

### Returns

The return value specifies the outcome of the escape. This value is positive if the escape is successful. Otherwise, it is negative.

## GETEXTENDEDTEXTMETRICS

short Escape(hdc, GETEXTENDEDTEXTMETRICS, sizeof(WORD), lpInData, lpOutData)

The GETEXTENDEDTEXTMETRICS printer escape fills the buffer pointed to by the lpOutData parameter with the extended text metrics for the selected font.

Parameter	Description
hdc	HDC Identifies the device context.
lpInData	<u>LPWORD</u> Points to an unsigned 16-bit integer that specifies the number of bytes pointed to by the lpOutData parameter.
lpOutData	<u>EXTTEXTMETRIC FAR *</u> Points to an EXTTEXTMETRIC structure. For more information about this structure, see the following Comments section.

### Returns

The return value specifies the number of bytes copied to the buffer pointed to by the lpOutData parameter. This value will never exceed that specified in the nSize member pointed to by the lpInData parameter. The return value is zero if the selected font does not have the extended text metrics or if the escape fails or is not implemented.

### Comments

The lpOutData parameter points to an EXTTEXTMETRIC structure, which has the following form:

```
struct EXTTEXTMETRIC {
    short etmSize;
    short etmPointSize;
    short etmOrientation;
    short etmMasterHeight;
    short etmMinScale;
    short etmMaxScale;
    short etmMasterUnits;
    short etmCapHeight;
    short etmXHeight;
    short etmLowerCaseAscent;
    short etmLowerCaseDescent;
    short etmSlant;
    short etmSuperScript;
    short etmSubScript;
    short etmSuperScriptSize;
    short etmSubScriptSize;
    short etmUnderlineOffset;
    short etmUnderlineWidth;
    short etmDoubleUpperUnderlineOffset;
    short etmDoubleLowerUnderlineOffset;
    short etmDoubleUpperUnderlineWidth;
    short etmDoubleLowerUnderlineWidth;
    short etmStrikeOutOffset;
    short etmStrikeOutWidth;
    WORD etmKernPairs;
    WORD etmKernTracks;
};
```

Following are the members in the EXTTEXTMETRIC structure:

etmSize	Specifies the size of the structure, in bytes.						
etmPointSize	Specifies the nominal point size of this font, in twips (1/20 of a point, or 1/1440 inch). This is the intended size of the font; the actual size may differ slightly depending on the resolution of the device.						
etmOrientation	Specifies the orientation of the font. The etmOrientation member may be any of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Either orientation</td></tr><tr><td>1</td><td>Portrait</td></tr></tbody></table>	Value	Meaning	0	Either orientation	1	Portrait
Value	Meaning						
0	Either orientation						
1	Portrait						

## 2 Landscape

These values refer to the ability of this font to be placed on a page with the given orientation. A portrait page has a height that is greater than its width. A landscape page has a width that is greater than its height.

etmMasterHeight	Specifies the font size, in device units, for which the values in this font's extent table are exact.
etmMinScale	Specifies the minimum valid size for this font. The following equation illustrates how the minimum point size is determined:  $\text{smallest point size} = (\text{etmMinScale} * 72) / \text{dfVertRes}$ <p>The value 72 represents the number of points per inch. The dfVertRes value is the number of dots per inch.</p>
etmMaxScale	Specifies the maximum valid size for this font. The following equation illustrates how the maximum point size is determined:  $\text{largest point size} = (\text{etmMaxScale} * 72) / \text{dfVertRes}$ <p>The value 72 represents the number of points per inch. The dfVertRes value is the number of dots per inch.</p>
etmMasterUnits	Specifies the integer number of units per em where an em equals the value of the etmMasterHeight member. (That is, etmMasterUnits is emtMasterHeight expressed in font units instead of device units.)
etmCapHeight	Specifies the height, in font units, of uppercase characters in the font. Typically, this is the height of capital H.
etmXHeight	Specifies the height, in font units, of lowercase characters in the font. Typically, this is the height of lowercase x.
etmLowerCaseAscent	Specifies the distance, in font units, that the ascender of lowercase letters extends above the base line. Typically, this is the height of lowercase d.
etmLowerCaseDescent	Specifies the distance, in font units, that the descender of lowercase letters extends below the base line. Typically, this is specified for the descender of lowercase p.
etmSlant	Specifies, for an italic or slanted font, the angle of the slant measured in tenths of a degree clockwise from the upright version of the font.
etmSuperScript	Specifies, in font units, the recommended amount to offset superscript characters from the base line. This is typically a negative value.
etmSubScript	Specifies, in font units, the recommended amount to offset subscript characters from the base line. This is typically a positive value.
etmSuperScriptSize	Specifies, in font units, the recommended size of superscript characters for this font.
etmSubScriptSize	Specifies, in font units, the recommended size of subscript characters for this font.
etmUnderlineOffset	Specifies, in font units, the offset downward from the base line where the top of a single underline bar should appear.
etmUnderlineWidth	Specifies, in font units, the thickness of the underline bar.
etmDoubleUpperUnderlineOffset	Specifies the offset, in font units, downward from the base line where the top of the upper double-underline bar should appear.
etmDoubleLowerUnderlineOffset	Specifies the offset, in font units, downward from the base line where the top of the lower double-underline bar should appear.

<code>etmDoubleUpperUnderlineWidth</code>	Specifies, in font units, the thickness of the upper underline bar.
<code>etmDoubleLowerUnderlineWidth</code>	Specifies, in font units, the thickness of the lower underline bar.
<code>etmStrikeOutOffset</code>	Specifies, in font units, the offset upward from the base line where the top of a strikeout bar should appear.
<code>etmStrikeOutWidth</code>	Specifies the thickness, in font units, of the strikeout bar.
<code>etmKernPairs</code>	Specifies the number of character kerning pairs defined for this font. An application can use this value to calculate the size of the pair-kern table returned by the <code>GETPAIRKERNTABLE</code> escape. It will not be greater than 512 kerning pairs.
<code>etmKernTracks</code>	Specifies the number of kerning tracks defined for this font. An application can use this value to calculate the size of the track-kern table returned by the <code>GETTRACKKERNTABLE</code> escape. It will not be greater than 16 kerning tracks.

The values returned in many of the members of the `EXTTEXTMETRIC` structure are affected by whether relative character widths are enabled or disabled. For more information, see the description of the `ENABLERELATIVEWIDTHS` escape earlier in this topic.

## GETEXTENTTABLE

short Escape(hdc, GETEXTENTTABLE, sizeof(CHAR\_RANGE\_STRUCT), lpInData, lpOutData)

The GETEXTENTTABLE printer escape retrieves the width (extent) of individual characters from a group of consecutive characters in the character set for the selected font.

Parameter	Description
hdc	HDC Identifies the device context.
lpInData	<u>LPSTR</u> Points to a CHAR_RANGE_STRUCT structure that defines the range of characters for which the width is to be retrieved. For more information about this structure, see the following Comments section.
lpOutData	<u>LPINT</u> Points to an array of short integers that receives the character widths. The size of the array must be at least (chLast - chFirst + 1).

### Returns

The return value specifies the outcome of the escape. This value is 1 if the escape is successful. It is zero if the escape is not successful. If the escape is not implemented, the return value is zero.

### Comments

The lpInData parameter points to a CHAR\_RANGE\_STRUCT structure that defines the range of characters for which the width is to be retrieved. This structure has the following form:

```
struct CHAR_RANGE_STRUCT {
    CHAR chFirst;
    CHAR chLast;
};
```

Following are the members in the CHAR\_RANGE\_STRUCT structure:

- chFirst Specifies the character code of the first character whose width is to be retrieved.
- chLast Specifies the character code of the last character whose width is to be retrieved.

How an application uses the retrieved values depends upon whether relative character widths are enabled or disabled. For more information, see the description of the ENABLERELATIVEWIDTHS escape, earlier in this topic.

## GETFACENAME

short Escape(hdc, GETFACENAME, NULL, NULL, lpFaceName)

The GETFACENAME printer escape retrieves the face name of the current physical font.

Parameter	Description
hdc	HDC Identifies the device context.
lpFaceName	<u>LPSTR</u> Points to a buffer of characters to receive the face name. This buffer must be at least 60 bytes in length.

### Returns

The return value is positive if the escape was successful. This value is zero if the escape is not implemented or negative if an error occurred.

### Comments

This escape is used only by PostScript printer drivers.



## GETPAIRKERNTABLE

short Escape(hdc, GETPAIRKERNTABLE, NULL, NULL, lpOutData)

The GETPAIRKERNTABLE printer escape fills the buffer pointed to by the lpOutData parameter with the character-pair kerning table for the selected font.

Parameter	Description
hdc	HDC Identifies the device context.
lpOutData	KERNPAIR FAR * Points to an array of KERNPAIR structures. This array must be large enough to accommodate the entire character-pair kerning table for the font. The number of character-kerning pairs in the font can be obtained from the EXTTEXTMETRIC structure returned by the GETEXTENDEDTEXTMETRICS escape. For more information about this structure, see the following Comments section.

### Returns

The return value specifies the number of KERNPAIR structures copied to the buffer. This value is zero if the font does not have kerning pairs defined or the escape fails or is not implemented.

### Comments

The KERNPAIR structure has the following form:

```
struct KERNPAIR {
    union {
        BYTE each [2]; /* 'each' and 'both' share same memory */
        WORD both;
    } kpPair;
    short kpKernAmount;
};
```

Following are the members in the KERNPAIR structure:

each	Specifies the character codes for the kerning pair.
both	Specifies a 16-bit value in which the first character in the kerning pair is in the low-order byte and the second character is in the high-order byte.
kpKernAmount	Specifies the signed amount that this pair will be kerned if they appear side by side in the same font and size. This value is typically negative because pair-kerning usually results in two characters being set tighter than normal.

The array of KERNPAIR structures is sorted in increasing order by the kpPair.both member.

The values returned in KERNPAIR structures are affected by whether relative character widths are enabled or disabled. For more information, see the description of the ENABLERELATIVEWIDTHS escape earlier in this topic.

## GETPHYSIZESIZE

short Escape(hdc, GETPHYSIZESIZE, NULL, NULL, lpDimensions)

The GETPHYSIZESIZE printer escape retrieves the physical page size and copies it to the location pointed to by the lpDimensions parameter.

Parameter	Description
hdc	HDC Identifies the device context.
lpDimensions	<u>LPPOINT</u> Points to a <u>POINT</u> structure that will receive the physical page dimensions (in the current orientation). The x member of the POINT structure receives the horizontal size, in device units, and the y member receives the vertical size, in device units.

### Returns

The return value specifies the outcome of the escape. This value is positive if the escape is successful. Otherwise, it is zero or negative.

## GETPRINTINGOFFSET

short Escape(hdc, GETPRINTINGOFFSET, NULL, NULL, lpOffset)

The GETPRINTINGOFFSET printer escape retrieves the offset from the upper-left corner of the physical page where the actual printing or drawing begins. This escape is generally not useful for devices that allow the user to set the origin of the printable area directly.

Parameter	Description
hdc	HDC Identifies the device context.
lpOffset	LPPOINT Points to a POINT structure that will receive the printing offset (in the current orientation). The x member of the POINT structure receives the horizontal coordinate of the printing offset, in device units, and the y member receives the vertical coordinate of the printing offset, in device units.

### Returns

The return value specifies the outcome of the escape. This value is positive if the escape is successful. Otherwise, it is zero or negative.

## GETSCALINGFACTOR

short Escape(hdc, GETSCALINGFACTOR, NULL, NULL, lpFactors)

The GETSCALINGFACTOR printer escape retrieves the scaling factors for the x-axis and y-axis of a printing device. For each scaling factor, the escape copies an exponent of 2 to the location pointed to by the lpFactors parameter. For example, the value 3 is copied to lpFactors if the scaling factor is 8.

Scaling factors are used by printing devices that support graphics at a smaller resolution than text.

Parameter	Description
hdc	HDC Identifies the device context.
lpFactors	<u>LPPOINT</u> Points to the <u>POINT</u> structure that will receive the scaling factor. The x member of the POINT structure receives the scaling factor for the x-axis and the y member receives the scaling factor for the y-axis.

### Returns

The return value specifies the outcome of the escape. This value is positive if the escape is successful. Otherwise, it is zero or negative.

## GETSETPAPERBINS

short Escape(hdc, GETSETPAPERBINS, nCount, lpInData, lpOutData)

The GETSETPAPERBINS printer escape is maintained for backwards compatibility. Applications written for Windows 3.1 should call the DeviceCapabilities function with the DC\_BINS index to retrieve the number of available paper bins and use the ExtDeviceMode function to set the current paper bin.

This escape retrieves the number of paper bins available on a printer and sets the current paper bin. For more information about actions performed by this escape, see the following Comments section.

Parameter	Description
hdc	HDC Identifies the device context.
nCount	int Specifies the number of bytes pointed to by the lpInData parameter.
lpInData	BinInfo FAR * Points to a BinInfo structure that specifies the new paper bin. It may be set to NULL. For more information about this structure, see the following Comments section.
lpOutData	BinInfo FAR * Points to a BinInfo structure that contains information about the current or previous paper bin and the number of bins available. For more information about this structure, see the following comments section.

### Returns

The return value is positive if the escape is successful. Otherwise, this value is zero or negative.

### Comments

There are three possible actions for this escape, depending on the values passed in the lpInData and lpOutData parameters:

lpInData	lpOutData	Action
NULL	BinInfo	Retrieves the number of bins and the number of the current bin.
BinInfo	BinInfo	Sets the current bin to the number specified in the BinNumber member of the structure to which the lpInData parameter points and retrieves the number of the previous bin.
BinInfo	NULL	Sets the current bin to the number specified in the BinNumber member of the structure to which the lpInData parameter points.

The BinInfo structure has the following form:

```
struct BinInfo {
    int BinNumber;
    int cBins;
    int Reserved;
    int Reserved;
    int Reserved;
    int Reserved;
};
```

Following are the members of the BinInfo structure:

BinNumber	Identifies the current or previous paper bin.
cBins	Specifies the number of paper bins available.

Once a new bin is set, the selection takes effect immediately; the next page printed comes from the new bin.

## GETSETPAPERMETRICS

short Escape(hdc, GETSETPAPERMETRICS, sizeof(RECT), lpNewPaper, lpPrevPaper)

The GETSETPAPERMETRICS printer escape sets the paper type according to the given paper metrics information. It also retrieves the paper metrics information for the current printer.

This escape is obsolete. Printer drivers written for Windows version 3.0 and later may not support this escape. Applications can use the DeviceCapabilities and ExtDeviceMode functions to achieve the same functionality.

This escape expects a RECT structure representing the imageable area of the physical page and assumes the origin is situated in the upper-left corner.

Parameter	Description
hdc	HDC Identifies the device context.
lpNewPaper	<u>LPRECT</u> Points to a <u>RECT</u> structure that defines the new imageable area.
lpPrevPaper	<u>LPRECT</u> Points to a <u>RECT</u> structure that receives the previous imageable area.

### Returns

The return value is positive if the escape is successful. The value is zero if the escape is not implemented and negative if an error occurs.

## GETSETPRINTORIENT

short Escape(hdc, GETSETPRINTORIENT, nCount, lpInData, NULL)

The GETSETPRINTORIENT printer escape returns or sets the current paper orientation. This escape is obsolete. Printer drivers written for Windows version 3.0 and later may not support this escape. An application should call the ExtDeviceMode function instead.

Parameter	Description
hdc	HDC Identifies the device context.
nCount	short Specifies the number of bytes pointed to by the lpInData parameter.
lpInData	ORIENT FAR * Points to an ORIENT structure that specifies the new paper orientation. For a description of this structure, see the following Comments section. It may be set to NULL, in which case the GETSETPRINTORIENT escape returns the current paper orientation.

### Returns

The return value specifies the current orientation if lpInData is NULL. Otherwise, this value is the previous orientation. This value is -1 if the escape fails.

### Comments

This escape is provided only for backward compatibility. New applications should use the graphics device interface (GDI) DeviceCapabilities and ExtDeviceMode functions instead.

The ORIENT structure has the following form:

```
struct ORIENT {
    DWORD Orientation;
    DWORD Reserved;
    DWORD Reserved;
    DWORD Reserved;
    DWORD Reserved;
};
```

The Orientation member can be one of these values:

Value	Meaning
1	New orientation is portrait.
2	New orientation is landscape.

## GETSETSCREENPARAMS

short Escape(hdc, GETSETSCREENPARAMS, sizeof(SCREENPARAMS), lpInData, lpOutData)

The GETSETSCREENPARAMS printer escape retrieves or sets the current screen information for rendering halftones.

Parameter	Description
hdc	HDC Identifies the device context.
lpInData	SCREENPARAMS FAR * Points to a SCREENPARAMS structure that contains the new screen information. For more information about this structure, see the following Comments section. This parameter may be NULL.
lpOutData	SCREENPARAMS FAR * Points to a SCREENPARAMS structure that retrieves the previous screen information. For more information about this structure, see the following Comments section. This parameter may be NULL.

### Returns

The return value specifies the outcome of the escape. This value is positive if the escape is successful. Otherwise, it is negative.

### Comments

This escape affects how device-independent bitmaps (DIBs) are rendered and how color objects are filled.

The SCREENPARAMS structure has the following form:

```
typedef struct tagSCREENPARAMS {  
    int    angle;  
    int    frequency;  
} SCREENPARAMS;
```

Following are the members of the SCREENPARAMS structure:

angle            Specifies, in degrees, the angle of the halftone screen.  
frequency        Specifies, in dots per inch, the screen frequency.



## GETTECHNOLOGY

short Escape(hdc, GETTECHNOLOGY, NULL, NULL, lpTechnology)

The GETTECHNOLOGY printer escape retrieves the general technology type for a printer, which allows an application to perform technology-specific actions.

Applications should avoid using this escape. Printer drivers written for Windows version 3.0 and later may not support this escape.

Parameter	Description
hdc	HDC Identifies the device context.
lpTechnology	<u>LPSTR</u> Points to a buffer to which the driver copies a null-terminated string containing the printer technology type, such as "PostScript".

### Returns

The return value specifies the outcome of the escape. This value is 1 if the escape is successful. It is zero if the escape is not successful or is not implemented.

## GETTRACKKERNTABLE

short Escape(hdc, GETTRACKKERNTABLE, NULL, NULL, lpOutData)

The GETTRACKKERNTABLE printer escape fills the buffer pointed to by the lpOutData parameter with the track-kerning table for the currently selected font.

Parameter	Description
hdc	HDC Identifies the device context.
lpOutdata	KERNTRACK FAR * Points to an array of KERNTRACK structures. This array must be large enough to accommodate all the kerning tracks for the font. The number of tracks in the font can be obtained from the EXTTEXTMETRIC structure which is returned by the GETEXTENDEDTEXTMETRICS escape. For more information about this structure, see the following Comments section.

### Returns

The return value specifies the number of KERNTRACK structures copied to the buffer. This value is zero if the font does not have kerning tracks defined or if the escape fails or is not implemented.

### Comments

The KERNTRACK structure has the following form:

```
struct KERNTRACK {
    short Degree;
    short MinSize;
    short MinAmount;
    short MaxSize;
    short MaxAmount;
};
```

Following are the members in the KERNTRACK structure:

Degree	Specifies the amount of track kerning. Increasingly negative values represent tighter track kerning, and increasingly positive values represent looser track kerning.
MinSize	Specifies, in device units, the minimum font size for which linear track kerning applies.
MinAmount	Specifies, in font units, the amount of track kerning to apply to font sizes less than or equal to the size specified by the MinSize member.
MaxSize	Specifies, in device units, the maximum font size for which linear track kerning applies.
MaxAmount	Specifies, in font units, the amount of track kerning to apply to font sizes greater than or equal to the size specified by the MaxSize member.

Between the MinSize and MaxSize font sizes, track kerning is a linear function from MinAmount to MaxAmount. The values returned in the KERNTRACK structures are affected by whether relative character widths are enabled or disabled. For more information, see the description of the ENABLERELATIVEWIDTHS escape earlier in this topic.

## GETVECTORBRUSHSIZE

short Escape(hdc, GETVECTORBRUSHSIZE, sizeof(LOGBRUSH), lpInData, lpOutData)

The GETVECTORBRUSHSIZE printer escape retrieves, in device units, the size of a plotter pen used to fill closed figures. Graphics device interface (GDI) uses this information to prevent the plotter pen from writing over the borders of the figure when filling closed figures.

Parameter	Description
hdc	HDC Identifies the device context.
lpInData	<u>LOGBRUSH FAR *</u> Points to a LOGBRUSH structure that specifies the brush for which data is to be returned.
lpOutData	<u>LPPOINT</u> Points to a <u>POINT</u> structure whose y member contains the width of the pen, in device units.

### Returns

The return value specifies the outcome of the escape. This value is 1 if the escape is successful. It is zero if the escape is not successful or is not implemented.

## GETVECTORPENSIZE

short Escape(hdc, GETVECTORPENSIZE, sizeof(LOGPEN), lpInData, lpOutData)

The GETVECTORPENSIZE printer escape retrieves the size, in device units, of a plotter pen. Graphics device interface (GDI) uses this information to prevent hatched brush patterns from overwriting the border of a closed figure.

Parameter	Description
hdc	HDC Identifies the device context.
lpInData	<u>LOGPEN FAR *</u> Points to a LOGPEN structure that specifies the pen for which the <u>width is to be retrieved</u> .
lpOutData	<u>LPPOINT</u> Points to a <u>POINT</u> structure that contains in its second word the width of the pen, in device units.

### Returns

The return value specifies the outcome of the escape. This value is 1 if the escape is successful and zero if the escape is not successful or if it is not implemented.

## MFCOMMENT

BOOL Escape(hdc, MFCOMMENT, nCount, lpComment, NULL)

The MFCOMMENT printer escape adds a comment to a metafile.

Parameter	Description
hdc	HDC Identifies the device context for the device on which the metafile appears.
nCount	short Specifies the number of characters in the string pointed to by the lpComment parameter.
lpComment	<u>LPSTR</u> Points to a string that contains the comment that will appear in the metafile.

### Returns

The return value specifies the outcome of the escape. This value is -1 if an error, such as insufficient memory or an invalid port specification, occurs. Otherwise, it is positive.

## MOUSETRAILS

short Escape(hdc, MOUSETRAILS, sizeof(WORD), lpTrailSize, NULL)

The MOUSETRAILS escape enables or disables mouse trails for display devices.

Parameter	Description												
hdc	HDC Identifies the device context.												
lpTrailSize	<p><u>LPINT</u> points to a 16-bit variable containing a value specifying the action to take and the number of mouse cursor images to display (trail size). The variable can be one of the following values:</p> <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>1 through 7</td><td>Enables mouse trails and sets the trail size to the specified number. A value of 1 requests a single mouse cursor. A value of 2 requests that one extra mouse cursor be drawn behind the current mouse cursor, and so on, up to a maximum of 7 total cursor images. The escape sets the MouseTrails entry in the WIN.INI file to the given value and returns the new trail size.</td></tr><tr><td>0</td><td>Disables mouse trails. The escape sets the MouseTrails entry to the negative value of the current trail size (if positive) and returns the negative value.</td></tr><tr><td>-1</td><td>Enables mouse trails. The display driver reads the MouseTrails entry from the [windows] section of the WIN.INI file. If the value of the entry is positive, the escape sets the trail size to the given value. If the entry is negative, the escape sets the trail size to the entry's absolute value and writes the positive value back to WIN.INI. If the MouseTrails entry is not found, the escape sets the trail size to 7 and writes a new MouseTrails entry to the WIN.INI file, setting its value to 7. The escape then returns the new trail size.</td></tr><tr><td>-2</td><td>Disables mouse trails but does not cause the display driver to update the WIN.INI file.</td></tr><tr><td>-3</td><td>Enables mouse trails but does not cause the display driver to update the WIN.INI file.</td></tr></tbody></table>	Value	Meaning	1 through 7	Enables mouse trails and sets the trail size to the specified number. A value of 1 requests a single mouse cursor. A value of 2 requests that one extra mouse cursor be drawn behind the current mouse cursor, and so on, up to a maximum of 7 total cursor images. The escape sets the MouseTrails entry in the WIN.INI file to the given value and returns the new trail size.	0	Disables mouse trails. The escape sets the MouseTrails entry to the negative value of the current trail size (if positive) and returns the negative value.	-1	Enables mouse trails. The display driver reads the MouseTrails entry from the [windows] section of the WIN.INI file. If the value of the entry is positive, the escape sets the trail size to the given value. If the entry is negative, the escape sets the trail size to the entry's absolute value and writes the positive value back to WIN.INI. If the MouseTrails entry is not found, the escape sets the trail size to 7 and writes a new MouseTrails entry to the WIN.INI file, setting its value to 7. The escape then returns the new trail size.	-2	Disables mouse trails but does not cause the display driver to update the WIN.INI file.	-3	Enables mouse trails but does not cause the display driver to update the WIN.INI file.
Value	Meaning												
1 through 7	Enables mouse trails and sets the trail size to the specified number. A value of 1 requests a single mouse cursor. A value of 2 requests that one extra mouse cursor be drawn behind the current mouse cursor, and so on, up to a maximum of 7 total cursor images. The escape sets the MouseTrails entry in the WIN.INI file to the given value and returns the new trail size.												
0	Disables mouse trails. The escape sets the MouseTrails entry to the negative value of the current trail size (if positive) and returns the negative value.												
-1	Enables mouse trails. The display driver reads the MouseTrails entry from the [windows] section of the WIN.INI file. If the value of the entry is positive, the escape sets the trail size to the given value. If the entry is negative, the escape sets the trail size to the entry's absolute value and writes the positive value back to WIN.INI. If the MouseTrails entry is not found, the escape sets the trail size to 7 and writes a new MouseTrails entry to the WIN.INI file, setting its value to 7. The escape then returns the new trail size.												
-2	Disables mouse trails but does not cause the display driver to update the WIN.INI file.												
-3	Enables mouse trails but does not cause the display driver to update the WIN.INI file.												

### Returns

The return value specifies the new trail size if the escape is successful. The return value is zero if the escape is not supported.

## NEWFRAME

short Escape(hdc, NEWFRAME, NULL, NULL, NULL)

The NEWFRAME printer escape is maintained for backwards compatibility. Applications written for Windows 3.1 should use the StartPage and EndPage functions.

This escape informs the device that the application has finished writing to a page. It is typically used with a printer to direct the device driver to advance to a new page.

Parameter	Description
hdc	HDC Identifies the device context.

### Returns

The return value specifies the outcome of the escape. This value is positive if the escape is successful. Otherwise, it is one of the following values:

Value	Meaning
SP_APPABORT	Job was terminated because the application's Abort function returned zero.
SP_ERROR	General error.
SP_OUTOFDISK	Not enough disk space is currently available for spooling, and no more space will become available.
SP_OUTOFMEMORY	Not enough memory is available for spooling.
SP_USERABORT	User terminated the job through Print Manager.

### Comments

Do not use the NEXTBAND escape with the NEWFRAME escape. For banding device drivers, graphics device interface (GDI) replays a metafile to the printer, simulating a sequence of NEXTBAND escapes.

The NEWFRAME escape restores the default values of the device context. Consequently, if a font other than the default font is selected when the application calls the NEWFRAME escape, the application must select the font again following the NEWFRAME escape.

The NEWFRAME escape should not be used inside metafiles.

## NEXTBAND

short Escape(hdc, NEXTBAND, NULL, NULL, lpBandRect)

The NEXTBAND printer escape informs the device driver that the application has finished writing to a band, causing the device driver to send the band to Print Manager and return the coordinates of the next band. Applications that process banding themselves use this escape.

Parameter	Description
hdc	HDC Identifies the device context.
lpBandRect	LPRECT Points to the <u>RECT</u> structure that will receive the next band coordinates. The device driver copies the <u>device</u> coordinates of the next band into this structure.

### Returns

The return value specifies the outcome of the escape. This value is positive if the escape is successful. A return value of zero indicates that an error occurred. In addition, the following error values are defined:

Value	Meaning
SP_APPABORT	Job was terminated because the application's Abort function returned zero.
SP_ERROR	General error.
SP_OUTOFDISK	Not enough disk space is currently available for spooling, and no more space will become available.
SP_OUTOFMEMORY	Not enough memory is available for spooling.
SP_USERABORT	User terminated the job through Print Manager.

### Comments

The NEXTBAND escape sets the band rectangle to the empty rectangle when printing reaches the end of a page.

Do not use the NEWFRAME escape with the NEXTBAND escape.

The NEXTBAND escape should not be used inside metafiles.



## PASSTHROUGH

short Escape(hdc, PASSTHROUGH, NULL, lpInData, NULL)

The PASSTHROUGH printer escape allows the application to send data directly to the printer, bypassing the standard print-driver code.

Note: To use this escape, an application must have complete information about how the particular printer operates.

Parameter	Description
hdc	HDC Identifies the device context.
lpInData	<u>LPSTR</u> Points to a structure whose first word (16 bits) contains the number of bytes of input data. The remaining bytes of the structure contain the data itself.

### Returns

The return value specifies the number of bytes transferred to the printer if the escape is successful. This value is less than or equal to zero if the escape is not successful or not implemented.

### Comments

There may be restrictions on the kinds of device data an application can send to the device without interfering with the operation of the driver. In general, applications must avoid resetting the printer or causing the page to be printed.

It is strongly recommended that applications do not perform actions that consume printer memory, such as downloading a font or a macro.

An application can avoid corrupting its data stream when issuing multiple, consecutive PASSTHROUGH escapes by not accessing the printer any other way during the sequence.

An application can guarantee that the PASSTHROUGH escape will be successful if it uses a "save" PostScript operator before sending PASSTHROUGH data and a "restore" operator after. Avoiding graphics device interface (GDI) functions between calls to the PASSTHROUGH escape and avoiding commands that cause a page to eject are other means to ensure that the escape will be successful.

## POSTSCRIPT\_DATA

The POSTSCRIPT\_DATA printer escape is identical to the PASSTHROUGH escape.

## POSTSCRIPT\_IGNORE

short Escape(hdc, POSTSCRIPT\_IGNORE, NULL, lpfOutput, NULL)

The POSTSCRIPT\_IGNORE printer escape sets a flag indicating whether or not to suppress output.

Parameter	Description
hdc	HDC Identifies the device context.
lpfOutput	<u>BOOL FAR*</u> Points to a flag indicating whether output should be suppressed. This value is nonzero to suppress output and zero otherwise.

### Returns

The return value specifies the previous setting of the output flag.

### Comments

Applications that generate their own PostScript code can use the POSTSCRIPT\_IGNORE escape to prevent the PostScript device driver from generating output.

## QUERYESCSUPPORT

short Escape(hdc, QUERYESCSUPPORT, sizeof(int), lpEscNum, NULL)

The QUERYESCSUPPORT printer escape determines whether a particular escape is implemented by the device driver.

Parameter	Description
hdc	HDC Identifies the device context.
lpEscNum	<u>LPINT</u> Points to a short integer that specifies the escape function to be checked.

### Returns

The return value specifies whether a particular escape is implemented. This value is nonzero for implemented escape functions. Otherwise, it is zero.

If the lpEscNum parameter is set to DRAWPATTERNRECT, the return value is one of the following values:

Value	Meaning
0	DRAWPATTERNRECT is not implemented.
1	DRAWPATTERNRECT is implemented for a printer other than the HP LaserJet IIP; this printer supports white rules.
2	DRAWPATTERNRECT is implemented for the HP LaserJet IIP.

## RESTORE\_CTM

short Escape(hdc, RESTORE\_CTM, NULL, NULL, NULL)

The RESTORE\_CTM printer escape restores the previously saved current transformation matrix.

The current transformation matrix controls the manner in which coordinates are translated, rotated, and scaled by the device. By using matrices, an application can combine these operations in any order to produce the desired mapping for a particular picture.

Parameter	Description
hdc	HDC Identifies the device context.

### Returns

The return value specifies the number of SAVE\_CTM escape calls without a corresponding RESTORE\_CTM call. The return value is -1 if the escape is unsuccessful.

### Comments

This escape is used only by PostScript printer drivers.

Applications should not make any assumptions about the initial contents of the current transformation matrix.

## SAVE\_CTM

short Escape(hdc, SAVE\_CTM, NULL, NULL, NULL)

The SAVE\_CTM printer escape saves the current transformation matrix.

The current transformation matrix controls the manner in which coordinates are translated, rotated, and scaled by the device. By using matrices, an application can combine these operations in any order to produce the desired mapping for a particular picture.

An application can restore the matrix by using the RESTORE\_CTM escape.

An application typically saves the current transformation matrix before changing it. This allows the application to restore the previous state upon completion of a particular operation.

Parameter	Description
hdc	HDC Identifies the device context.

### Returns

The return value specifies the number of SAVE\_CTM escape calls without a corresponding RESTORE\_CTM call. The return value is zero if the escape is unsuccessful.

### Comments

This escape is used only by PostScript printer drivers.

Applications should not make any assumptions about the initial contents of the current transformation matrix.

Applications are expected to restore the contents of the current transformation matrix.

## SELECTPAPERSOURCE

The SELECTPAPERSOURCE printer escape has been superseded by the DeviceCapabilities function (using the DC\_BINS value). SELECTPAPERSOURCE is provided only for backward compatibility.

## SETABORTPROC

short Escape(hdc, SETABORTPROC, NULL, lpAbortFunc, NULL)

The SETABORTPROC printer escape is maintained for backwards compatibility. Applications written for Windows 3.1 should use the SetAbortProc function.

This escape sets the Abort function for the print job.

To allow a print job to be canceled during spooling, an application must set the Abort function before the print job is started with the STARTDOC escape. Print Manager calls the Abort function during spooling to allow the application to cancel the print job or to take appropriate action for such errors as running out of disk space. If no Abort function is set, the print job will fail if there is not enough disk space for spooling.

Parameter	Description
hdc	HDC Identifies the device context.
lpAbortFunc	FARPROC Points to the application-supplied Abort function. For details, see the following Comments section.

### Returns

The return value specifies the outcome of the escape. This value is greater than zero if the escape is successful. Otherwise, it is less than zero.

### Comments

The address passed as the lpAbortFunc parameter must be created by using the MakeProcInstance function.

The callback function must use the Pascal calling convention and must be declared FAR. The Abort function must have the following form:

```
short FAR PASCAL AbortFunc(hPr,code)
HDC hPr;
short code;
```

AbortFunc is a placeholder for the application-supplied function name. The actual name must be exported by including it in an EXPORTS statement in the module-definition (.DEF) file for the application.

Following are the parameters in the Abort function:

hPr	Identifies the device context.
code	Specifies whether an error has occurred. This parameter is zero if no error has occurred. It is <u>SP_OUTOFDISK</u> if Print Manager is currently out of disk space and more disk space will become available if the application waits. If code is <u>SP_OUTOFDISK</u> , the application does not have to abort the print job. If it does not abort the print job, it must yield to Print Manager by calling the <u>PeekMessage</u> or <u>GetMessage</u> function.

### Returns

The return value should be nonzero if the print job is to continue and zero if it is canceled.



## SETALLJUSTVALUES

short Escape(hdc, SETALLJUSTVALUES, sizeof(EXTTEXTDATA), lpInData, NULL)

The SETALLJUSTVALUES printer escape is not recommended. Applications should use the ExtTextOut function instead of this escape. This escape sets all of the text-justification values that are used for text output in Windows 3.0 and earlier.

Text justification is the process of inserting extra pixels among break characters in a line of text. The space character is normally used as a break character.

Parameter	Description
hdc	HDC Identifies the device context.
lpInData	EXTTEXTDATA FAR * Points to an EXTTEXTDATA structure that defines the text-justification values. For more information about this structure, see the Comments section.

### Returns

The return value specifies the outcome of the escape. This value is 1 if the escape is successful. Otherwise, it is zero.

### Comments

The lpInData parameter points to an EXTTEXTDATA structure that describes the text-justification values used for text output. The EXTTEXTDATA structure has the following form:

```
typedef struct {
    short nSize;
    LPALLJUSTREC lpInData;
    LPFONTINFO lpFont;
    LPTEXTXFORM lpXForm;
    LPDRAWMODE lpDrawMode;
} EXTTEXTDATA;
```

This structure contains a JUST\_VALUE\_STRUCT structure that has the following form:

```
typedef struct {
    short nCharExtra;
    WORD cch;
    short nBreakExtra;
    WORD nBreakCount;
} JUST_VALUE_STRUCT;
```

Following are the members of JUST\_VALUE\_STRUCT structure:

nCharExtra	Specifies the total extra space, in font units, that must be distributed over cch characters.
cch	Specifies the number of characters over which the nCharExtra member is distributed.
nBreakExtra	Specifies the total extra space, in font units, that is distributed over nBreakCount characters.
nBreakCount	Specifies the number of break characters over which the nBreakExtra member is distributed.

The units used for the nCharExtra and nBreakExtra members are the font units of the device and are dependent on whether relative character widths were enabled with the ENABLERELATIVEWIDTHS escape.

The values set with this escape apply to subsequent calls to the TextOut function. The driver stops distributing the extra space specified in the nCharExtra member when it has output the number of characters specified in the nCharCount member. Likewise, it stops distributing the space specified by the nBreakExtra member when it has output the number of characters specified by the nBreakCount member. A call on the same string to the GetTextExtent function made immediately after the call to the TextOut function will be processed in the same manner.

To reenble justification with the SetTextJustification and SetTextCharacterExtra functions, an application should call the SETALLJUSTVALUES escape and set the nCharExtra and nBreakExtra members to zero.

## SET\_ARC\_DIRECTION

short Escape(hdc, SET\_ARC\_DIRECTION, sizeof(int), lpDirection, NULL)

The SET\_ARC\_DIRECTION printer escape specifies the direction in which elliptical arcs are drawn using the graphics device interface (GDI) Arc function.

By convention, elliptical arcs are drawn counterclockwise by GDI. This escape lets an application draw paths containing arcs drawn clockwise.

Parameter	Description
hdc	HDC Identifies the device context.
lpDirection	<u>LPINT</u> Points to a short integer specifying the arc direction. It can be one of the following values:  COUNTERCLOCKWISE (0) CLOCKWISE (1)

### Returns

The return value is the previous arc direction.

### Comments

This escape maps to PostScript language elements and is intended for PostScript line devices.

## SET\_BACKGROUND\_COLOR

short Escape(hdc, SET\_BACKGROUND\_COLOR, nCount, lpNewColor, lpOldColor)

The SET\_BACKGROUND\_COLOR printer escape sets and retrieves the current background color for the device.

The background color is the color of the screen surface before an application draws anything on the device. This escape is particularly useful for color printers and film recorders.

This escape should be sent before the application draws anything on the current page.

Parameter	Description
hdc	HDC Identifies the device context.
nCount	int Specifies the number of bytes pointed to by the lpNewColor parameter.
lpNewColor	LPDWORD Points to a 32-bit integer specifying the desired background color. This parameter can be NULL if the application is merely retrieving the current background color.
lpOldColor	LPDWORD Points to a 32-bit integer that receives the previous background color. This parameter can be NULL if the application does not use the previous background color.

### Returns

The return value is nonzero if the escape is successful. This value is zero if it is unsuccessful.

### Comments

The default background color is white.

The background color is reset to the default when the device driver receives an ENDDOC or ABORTDOC escape.

## SET\_BOUNDS

short Escape(hdc, SET\_BOUNDS, sizeof(RECT), lpInData, NULL)

The SET\_BOUNDS printer escape sets the bounding rectangle for the picture being produced by the device driver supporting the given device context. This escape is used when creating images in a file format such as Encapsulated PostScript (EPS) and Hewlett-Packard Graphics Language (HPGL) for which there is a device driver.

Parameter	Description
hdc	HDC Identifies the device context.
lpInData	LPRECT Points to a RECT structure that specifies in device coordinates a rectangle that bounds the image to be output.

### Returns

The return value is nonzero if the escape was successful. Otherwise, it is zero.

### Comments

An application should issue this escape before each page in the image. For single-page images, this escape should be issued immediately before the STARTDOC escape.

When an application uses coordinate-transformation escapes, device drivers may not perform bounding box calculations correctly. When an application uses the SET\_BOUNDS escape, the driver does not have to calculate the bounding box.

Applications should always use this escape to ensure support for the Encapsulated PostScript (EPS) printing capabilities.

## SET\_CLIP\_BOX

short Escape(hdc, SET\_CLIP\_BOX, sizeof(RECT), lpInData, (LPSTR) NULL)

The SET\_CLIP\_BOX printer escape sets the clipping rectangle or restores the previous clipping rectangle. This escape is implemented by printer drivers that use the coordinate-transformation escapes TRANSFORM\_CTM, SAVE\_CTM, and RESTORE\_CTM.

When an application calls a graphics device interface (GDI) output function, GDI calculates a clipping rectangle bounding the primitive and passes both the primitive and the clipping rectangle to the printer driver. The printer driver is expected to clip the primitive to the specified bounding rectangle. However, when an application uses the coordinate-transformation escapes, the clipping rectangle calculated by GDI is usually invalid. An application can use the SET\_CLIP\_BOX escape to specify the correct clipping rectangle when coordinate transformations are used.

Parameter	Description
hdc	HDC Identifies the device context.
lpClipBox	LPRECT Points to a RECT structure containing the bounding rectangle of the clipping region. If lpClipBox is not NULL, the previous clipping rectangle is saved and the current clipping rectangle is set to the specified bounds. If lpClipBox is NULL, the previous clipping rectangle is restored.

### Returns

The return value is nonzero if the clipping rectangle was properly set. Otherwise, it is zero.

### Comments

This escape is used only by PostScript printer drivers.

## SETCOLORTABLE

short Escape(hdc, SETCOLORTABLE, sizeof(COLORTABLE\_STRUCT), lpInData, lpColor)

The SETCOLORTABLE printer escape sets an RGB color-table entry. If the device cannot supply the exact color, the function sets the entry to the closest possible approximation of the color.

Parameter	Description
hdc	HDC Identifies the device context.
lpInData	COLORTABLE_STRUCT FAR * Points to a structure that contains the index and <u>RGB</u> value of the color-table entry. For more information about the COLORTABLE_STRUCT structure, see the following Comments section.
lpColor	LPDWORD Points to the long integer that is to receive the <u>RGB</u> color value selected by the device driver to represent the requested color value.

### Returns

The return value specifies the outcome of the escape. This value is positive if the escape is successful. Otherwise, it is negative.

### Comments

The COLORTABLE\_STRUCT structure has the following form:

```
struct COLORTABLE_STRUCT {
    WORD  Index;
    DWORD rgb;
};
```

Following are the members of the COLORTABLE\_STRUCT structure:

Index	Specifies the color-table index. Color-table entries start at zero for the first entry.
rgb	Specifies the desired <u>RGB</u> color value.

The color table for a device is a shared resource; changing the system display color for one window changes it for all windows. Only application developers who have a thorough knowledge of the display driver should use this escape.

The SETCOLORTABLE escape has no effect on devices with fixed color tables.

This escape is intended for use by both printer and display drivers. However, the EGA and VGA color drivers do not support it.

This escape changes the palette used by the display driver. However, because the color-mapping algorithms for the driver will probably no longer work with a different palette, an extension has been added to this function.

If the color index pointed to by the lpInData parameter is 0xFFFF, the driver is to leave all color-mapping functionality to the calling application. The application must use the proper color-mapping algorithm and take responsibility for passing the correctly mapped physical color to the driver (instead of the logical RGB color) in such device-driver functions as RealizeObject and ColorInfo.

For example, if the device supports 256 colors with palette indexes of 0 through 255, an application determines which index contains the color that it wants to use in a certain brush. It then passes this index in the low-order byte of the doubleword logical color passed to the RealizeObject device-driver function. The driver uses this color exactly as passed instead of performing its usual color-mapping algorithm. If the application wants to reactivate the driver's color-mapping algorithm (that is, if it restores the original palette when switching from its window context), then the color index pointed to by lpInData should be 0xFFFFE.

## SETCOPYCOUNT

short Escape(hdc, SETCOPYCOUNT, sizeof(int), lpNumCopies, lpActualCopies)

The SETCOPYCOUNT printer escape is maintained for backwards compatibility. Applications written for Windows 3.1 should use the ExtDeviceMode function.

This escape specifies the number of uncollated copies of each page that the printer is to print.

Parameter	Description
hdc	HDC Identifies the device context.
lpNumCopies	<u>LPINT</u> Points to a short integer that contains the number of uncollated copies to be printed.
lpActualCopies	<u>LPINT</u> Points to a short integer that will receive the number of copies to be printed. This may be less than the number requested if the requested number is greater than the maximum copy count for the device.

### Returns

The return value specifies the outcome of the escape. It is 1 if the escape is successful and zero if the escape is not successful. The return value is zero if the escape is not implemented.

## SETKERNTRACK

short Escape(hdc, SETKERNTRACK, sizeof(int), lpNewTrack, lpOldTrack)

The SETKERNTRACK printer escape specifies which kerning track to use for drivers that support automatic track kerning. A kerning track of zero disables automatic track kerning.

When track kerning is enabled, the driver will automatically kern all characters according to the specified track. The driver will reflect this kerning both on the printer and in GetTextExtent function calls.

Parameter	Description
hdc	HDC Identifies the device context.
lpNewTrack	<u>LPINT</u> Points to a short integer that specifies the kerning track to use. A value of zero disables this feature. Values in the range 1 through the value of the <code>etmKernTracks</code> member correspond to positions in the track-kerning table (using 1 as the first item in the table). For more information, see the description of the <code>EXTTEXTMETRIC</code> structure provided in the description of the <code>GETEXTENDEDTEXTMETRICS</code> escape earlier in this topic.
lpOldTrack	<u>LPINT</u> Points to a short integer that will receive the previous kerning track.

### Returns

The return value specifies the outcome of the escape. It is 1 if the escape is successful and zero if the escape is not successful or not implemented.

### Comments

Automatic track kerning is disabled by default.

A driver does not have to support the SETKERNTRACK escape just because it supplies the track-kerning table to the application by using the `GETTRACKKERNTABLE` escape. In a case where `GETTRACKKERNTABLE` is supported but the SETKERNTRACK escape is not, the application must properly space the characters on the output device.



## SETLINECAP

short Escape(hdc, SETLINECAP, sizeof(int), lpNewCap, lpOldCap)

The SETLINECAP printer escape sets the line end cap.

A line end cap is that portion of a line segment that appears on either end of the segment. The cap may be square or circular. It can extend past or remain flush with the specified segment endpoints.

Parameter	Description										
hdc	HDC Identifies the device context.										
lpNewCap	<u>LPINT</u> Points to a short integer that specifies the end-cap type. Following are the possible values and their meanings: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>-1</td><td>Line segments are drawn by using the default graphics device interface (GDI) end cap.</td></tr><tr><td>0</td><td>Line segments are drawn with a squared end point that does not project past the specified segment length.</td></tr><tr><td>1</td><td>Line segments are drawn with a rounded end point; the diameter of this semicircular arc is equal to the line width.</td></tr><tr><td>2</td><td>Line segments are drawn with a squared end point that projects past the specified segment length. The projection is equal to half the line width.</td></tr></tbody></table>	Value	Meaning	-1	Line segments are drawn by using the default graphics device interface (GDI) end cap.	0	Line segments are drawn with a squared end point that does not project past the specified segment length.	1	Line segments are drawn with a rounded end point; the diameter of this semicircular arc is equal to the line width.	2	Line segments are drawn with a squared end point that projects past the specified segment length. The projection is equal to half the line width.
Value	Meaning										
-1	Line segments are drawn by using the default graphics device interface (GDI) end cap.										
0	Line segments are drawn with a squared end point that does not project past the specified segment length.										
1	Line segments are drawn with a rounded end point; the diameter of this semicircular arc is equal to the line width.										
2	Line segments are drawn with a squared end point that projects past the specified segment length. The projection is equal to half the line width.										
lpOldCap	<u>LPINT</u> Points to a short integer that specifies the previous end-cap setting.										

### Returns

The return value specifies the outcome of the escape. It is positive if the escape is successful. Otherwise, it is negative.

### Comments

This escape is used only by PostScript printer drivers.

The interpretation of this escape varies with page-description languages (PDLs). For its exact meaning, consult the PDL documentation.

This escape is also known as SETENDCAP.

## SETLINEJOIN

short Escape(hdc, SETLINEJOIN, sizeof(int), lpNewJoin, lpOldJoin)

The SETLINEJOIN printer escape specifies how a device driver will join two intersecting line segments. The intersection can form a rounded, squared, or mitered corner.

Parameter	Description										
hdc	HDC Identifies the device context.										
lpNewJoin	<u>LPINT</u> Points to a short integer that specifies the type of intersection. Following are the possible values and their meanings: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>-1</td><td>Line segments are joined by using the default graphics device interface (GDI) setting.</td></tr><tr><td>0</td><td>Line segments are joined with a mitered corner; the outer edges of the lines extend until they meet at an angle. This is referred to as a miter join.</td></tr><tr><td>1</td><td>Line segments are joined with a rounded corner; a semicircular arc with a diameter equal to the line width is drawn around the point where the lines meet. This is referred to as a round join.</td></tr><tr><td>2</td><td>Line segments are joined with a squared end point; the outer edges of the lines are not extended. This is referred to as a bevel join.</td></tr></tbody></table>	Value	Meaning	-1	Line segments are joined by using the default graphics device interface (GDI) setting.	0	Line segments are joined with a mitered corner; the outer edges of the lines extend until they meet at an angle. This is referred to as a miter join.	1	Line segments are joined with a rounded corner; a semicircular arc with a diameter equal to the line width is drawn around the point where the lines meet. This is referred to as a round join.	2	Line segments are joined with a squared end point; the outer edges of the lines are not extended. This is referred to as a bevel join.
Value	Meaning										
-1	Line segments are joined by using the default graphics device interface (GDI) setting.										
0	Line segments are joined with a mitered corner; the outer edges of the lines extend until they meet at an angle. This is referred to as a miter join.										
1	Line segments are joined with a rounded corner; a semicircular arc with a diameter equal to the line width is drawn around the point where the lines meet. This is referred to as a round join.										
2	Line segments are joined with a squared end point; the outer edges of the lines are not extended. This is referred to as a bevel join.										
lpOldJoin	<u>LPINT</u> Points to a short integer that specifies the previous line join setting.										

### Returns

The return value specifies the outcome of the escape. It is positive if the escape is successful. Otherwise, it is negative.

### Comments

This escape is used only by PostScript printer drivers.

The interpretation of this escape varies with page-description languages (PDLs). For its exact meaning, consult the PDL documentation.

If an application specifies a miter join but the angle of intersection is too small, the device driver ignores the miter setting and uses a bevel join instead.

## SETMITERLIMIT

short Escape(hdc, SETMITERLIMIT, sizeof(int), lpNewMiter, lpOldMiter)

The SETMITERLIMIT printer escape sets the miter limit for a device. The miter limit controls the angle at which a device driver replaces a miter join with a bevel join.

Parameter	Description
hdc	HDC Identifies the device context.
lpNewMiter	<u>LPINT</u> Points to a short integer that specifies the desired miter limit. Only values greater than or equal to -1 are valid. If the value is -1, the driver will use the default graphics device interface (GDI) miter limit.
lpOldMiter	<u>LPINT</u> Points to a short integer that specifies the previous miter-limit setting.

### Returns

The return value specifies the outcome of the escape. This value is positive if the escape is successful. Otherwise, it is negative.

### Comments

This escape is used only by PostScript printer drivers.

The miter limit is defined as follows:

$$\text{miter length} / \text{line width} = 1 / \sin(x/2)$$

where x is the angle of the line join, in radians.

The interpretation of this escape varies with page-description languages (PDLs). For its exact meaning, consult the PDL documentation.

## SET\_POLY\_MODE

short Escape(hdc, SET\_POLY\_MODE, sizeof(int), lpMode, NULL)

The SET\_POLY\_MODE printer escape sets the poly mode for the device driver. The poly mode is a state variable indicating how to interpret calls to graphics device interface (GDI) Polygon and Polyline functions.

The SET\_POLY\_MODE escape enables a device driver to draw shapes (such as Bezier curves) not directly supported by GDI. This permits applications that draw complex curves to send the curve description directly to a device without having to simulate the curve as a polygon with a large number of points.

Parameter	Description
hdc	HDC Identifies the device context.
lpMode	<u>LPINT</u> Points to a short integer specifying the desired poly mode. The poly mode is a state variable indicating how points in <u>Polygon</u> or <u>Polyline</u> function calls should be interpreted. Device drivers are not required to support all possible modes. A device driver returns zero if it does not support the specified mode. The lpMode parameter may be one of the following values:

Value	Meaning
PM_POLYLINE (1)	Points define a conventional polygon or polyline.
PM_BEZIER (2)	Points define a sequence of 4-point Bezier spline curves. The first curve passes through the first four points, with the first and fourth points as endpoints and the second and third points as control points. Each subsequent curve in the sequence has the endpoint of the previous curve as its start point, the next two points as control points, and the third as its endpoint. The last curve in the sequence is permitted to have fewer than four points. If the curve has only one point, it is considered a point. If it has two points, it is a line segment. If it has three points, it is a parabola defined by drawing a Bezier curve with the first and third points as endpoints and the two control points equal to the second point.
PM_POLYLINESEGMENT (3)	Points specify a list of coordinate pairs. Line segments are drawn connecting each successive pair of points.
PM_POLYSCANLINE (4)	Points specify a list of coordinate pairs. Line segments are drawn connecting each successive pair of points. Each line segment is a nominal-width line drawn with the current brush. Each line segment must be strictly vertical or horizontal, and scan lines must be passed in strictly increasing or decreasing order. This mode is only used for polygon calls.

### Returns

The return value is the previous poly mode. If the return value is zero, the device driver did not handle the request.

### Comments

This escape is used only by PostScript printer drivers.

An application should issue the SET\_POLY\_MODE escape before it draws a complex curve. It should then call the Polyline or Polygon function with the desired control points defining the curve. After drawing the curve, the application should reset the driver to its previous state by issuing the SET\_POLY\_MODE escape.

Polyline calls draw using the currently selected pen.

Polygon calls draw using the currently selected pen and brush. If the start point and endpoint are not equal, a line is drawn from the start point to the endpoint before the polygon (or curve) is filled.

GDI treats Polygon calls using PM\_POLYLINESEGMENT mode exactly the same as Polyline calls.

Four points define a Bezier curve. GDI generates the curve by connecting the first and second, second and third, and third and fourth points. GDI then connects the midpoints of these consecutive line segments. Finally, GDI connects the midpoints of the lines connecting the midpoints, and so forth.

The line segments drawn in this way converge to a curve defined by the following parametric equations, expressed as a function of the independent variable t.

$$X(t) = (1-t)^3 x_1 + 3(1-t)^2 t x_2 + 3(1-t)t^2 x_3 + t^3 x_4$$

$$Y(t) = (1-t)^3 y_1 + 3(1-t)^2 t y_2 + 3(1-t)t^2 y_3 + t^3 y_4$$

The points (x1, y1), (x2, y2), (x3, y3) and (x4, y4) are the control points defining the curve. The independent variable t varies from 0 to 1.

Primitive types other than PM\_BEZIER and PM\_POLYLINESEGMENT may be added to this escape in the future. Applications should check the return value from this escape to determine whether the driver supports the specified poly mode.

## SET\_SCREEN\_ANGLE

short Escape(hdc, SET\_SCREEN\_ANGLE, sizeof(int), lpAngle, NULL)

The SET\_SCREEN\_ANGLE printer escape sets the current screen angle to the desired angle and enables an application to simulate the tilting of a photographic mask in producing a color separation for a particular primary.

Parameter	Description
hdc	HDC Identifies the device context.
lpAngle	LPINT Points to a short integer specifying the desired screen angle in tenths of a degree. The angle is measured counterclockwise.

### Returns

The return value is the previous screen angle.

### Comments

Four-color process separation is the process of separating the colors comprising an image into four component primaries: cyan, magenta, yellow, and black. The image is then reproduced by overprinting each primary.

In traditional four-color process printing, half-tone images for each of the four primaries are photographed against a mask tilted to a particular angle. Tilting the mask in this manner minimizes unwanted moiré patterns caused by overprinting two or more colors.

The device driver defines the default screen angle.

## SET\_SPREAD

short Escape(hdc, SET\_SPREAD, sizeof(int), lpSpread, NULL)

The SET\_SPREAD printer escape sets the amount that nonwhite primitives are expanded for a given device to provide a slight overlap between primitives to compensate for imperfections in the reproduction process.

Spot color separation is the process of separating an image into each distinct color used in the image. The image is reproduced by overprinting each successive color in the image.

When reproducing a spot-separated image, the printing equipment must be calibrated to align each page exactly on each pass. However, differences in temperature, humidity, and so forth between passes often cause images to align imperfectly on subsequent passes. For this reason, lines in spot separations are often widened (spread) slightly to make up for problems in registering subsequent passes through the printer. This process is called trapping. The SET\_SPREAD escape implements this process.

Parameter	Description
hdc	HDC Identifies the device context.
lpSpread	<u>LPINT</u> Points to a short integer that specifies the amount, in pixels, by which all nonwhite primitives are to be expanded.

### Returns

The return value is the previous spread value.

### Comments

The default spread is zero.

The current spread applies to all bordered primitives (whether or not the border is visible) and text.

## STARTDOC

short Escape(hdc, STARTDOC, nCount, lpDocName, NULL)

The STARTDOC printer escape is maintained for backwards compatibility. Applications written for Windows 3.1 should use the StartDoc function.

This escape informs the device driver that a new print job is starting and that all subsequent NEWFRAME escape calls should be spooled under the same job until an ENDDOC escape call occurs. This ensures that documents longer than one page will not be interspersed with other jobs.

Parameter	Description
hdc	HDC Identifies the device context.
nCount	short Specifies the number of characters in the string pointed to by the lpDocName parameter.
lpDocName	LPSTR Points to a null-terminated string that specifies the name of the document. The document name is displayed in the Print Manager window. The maximum length of this string is 31 characters plus the terminating null character.

### Returns

The return value specifies the outcome of the escape. It is -1 if an error such as insufficient memory or an invalid port specification occurs. Otherwise, it is positive.

### Comments

Following is the correct sequence of events in a printing operation:

- 1 Create the device context.
- 2 Set the Abort function to keep out-of-disk-space errors from terminating a printing operation.  
An Abort procedure that handles these errors must be set by using the SETABORTPROC escape.
- 3 Begin the printing operation with the STARTDOC escape.
- 4 Begin each new page with the NEWFRAME escape or each new band with the NEXTBAND escape.
- 5 End the printing operation with the ENDDOC escape.
- 6 Destroy the Cancel dialog box, if any.
- 7 Free the procedure-instance address of the Abort function.

If an application encounters a printing error or a canceled print operation, it must not attempt to terminate the operation by using the Escape function with either the ENDDOC or ABORTDOC escape. Graphics device interface (GDI) automatically terminates the operation before returning the error value.

The STARTDOC escape should not be used inside metafiles.



## STRETCHBLT

The STRETCHBLT printer escape is provided for backwards compatibility. Applications should use the StretchBlt function instead of this escape.

See Also  
[StretchBlt](#)

## TRANSFORM\_CTM

short Escape(hdc, TRANSFORM\_CTM, 36, lpMatrix, NULL)

The TRANSFORM\_CTM printer escape modifies the current transformation matrix. The current transformation matrix controls the manner in which coordinates are translated, rotated, and scaled by the device. By using matrices, you can combine these operations in any order to produce the desired mapping for a particular picture.

The new current transformation matrix will contain the product of the matrix referenced by the lpMatrix parameter and the previous current transformation matrix ( $CTM = M * CTM$ ).

Parameter	Description
hdc	HDC Identifies the device context.
lpMatrix	LPSTR Points to a 3-by-3 array of 32-bit integer values specifying the new transformation matrix. Entries in the matrix are scaled to represent fixed-point real numbers. Each matrix entry is scaled by 65,536. The high-order word of the entry contains the whole integer portion, and the low-order word contains the fractional portion.

### Returns

The return value is nonzero if the escape was successful and zero if it was unsuccessful.

### Comments

This escape is used only by PostScript printer drivers.

When an application modifies the current transformation matrix, it must specify the clipping rectangle by issuing the SET\_CLIP\_BOX escape.

Applications should not make any assumptions about the initial value of the current transformation matrix.

## Printer Escapes

<u>ABORTDOC</u>	Superseded: use AbortDoc function
<u>BANDINFO</u>	Obsolete in Windows 3.1
<u>BEGIN_PATH</u>	Opens a path
<u>CLIP_TO_PATH</u>	Defines clip region bounded by path
<u>DEVICEDATA</u>	Same as PASSTHROUGH escape
<u>DRAFTMODE</u>	Superseded: Use DEVMODE structure
<u>DRAWPATTERNRECT</u>	Creates pattern on PCL printers
<u>ENABLEDUPLEX</u>	Superseded: use DEVMODE structure
<u>ENABLEPAIRKERNING</u>	Enables or disables kerning
<u>ENABLERELATIVewidthS</u>	Enables or disables relative char widths
<u>ENDDOC</u>	Superseded: Use EndDoc function
<u>END_PATH</u>	Ends a path
<u>ENUMPAPERBINS</u>	Superseded: Use DeviceCapabilities function
<u>ENUMPAPERMETRICS</u>	Superseded: Use DeviceCapabilities function
<u>EPSPRINTING</u>	Allows EPS printing only
<u>EXT_DEVICE_CAPS</u>	Superseded: Use GetDeviceCaps function
<u>EXTTEXTOUT</u>	Superseded: Use ExtTextOut function
<u>FLUSHOUTPUT</u>	Obsolete in Windows 3.1
<u>GETCOLORTABLE</u>	Obsolete in Windows 3.1
<u>GETEXTENDEDTEXTMETRICS</u>	Gets extended text metrics
<u>GETTEXTENTTABLE</u>	Superseded: Use GetCharWidth function
<u>GETFACENAME</u>	Gets face name of current font
<u>GETPAIRKERNTABLE</u>	Gets kerning-pair structures
<u>GETPHYSPAGE SIZE</u>	Gets size of physical page
<u>GETPRINTINGOFFSET</u>	Gets offset where printing starts
<u>GETSCALINGFACTOR</u>	Gets scaling factors for printer
<u>GETSETPAPERBINS</u>	Superseded: Use DeviceCapabilities function
<u>GETSETPAPERMETRICS</u>	Superseded: Use ExtDeviceMode function
<u>GETSETPRINTORIENT</u>	Superseded: Use ExtDeviceMode function
<u>GETSETSCREENPARAMS</u>	Gets or sets halftoning parameters
<u>GETTECHNOLOGY</u>	Gets technology type
<u>GETTRACKKERNTABLE</u>	Gets track-kerning table
<u>GETVECTORBRUSHSIZE</u>	Gets size of plotter brush
<u>GETVECTORPEN SIZE</u>	Gets size of plotter pen
<u>MFCOMMENT</u>	Adds comment to metafile
<u>MOUSETRAILS</u>	Enables or disables mouse trails
<u>NEWFRAME</u>	Superseded: Use StartPage and EndPage functions
<u>NEXTBAND</u>	Finished band, get next band
<u>PASSTHROUGH</u>	Sends data directly to printer
<u>POSTSCRIPT_DATA</u>	Same as PASSTHROUGH escape
<u>POSTSCRIPT_IGNORE</u>	Flag for suppressing output
<u>QUERYYESCSUPPORT</u>	Queries whether escape is supported
<u>RESTORE_CTM</u>	Restores current transformation matrix
<u>SAVE_CTM</u>	Saves current transformation matrix
<u>SELECTPAPERSOURCE</u>	Superseded: Use DeviceCapabilities function
<u>SETABORTPROC</u>	Superseded: Use SetAbortProc function
<u>SETALLJUSTVALUES</u>	Superseded: Use ExtTextOut function
<u>SET_ARC_DIRECTION</u>	Sets arc-drawing direction
<u>SET_BACKGROUND_COLOR</u>	Sets and gets background color
<u>SET_BOUNDS</u>	Sets bounding rectangle
<u>SET_CLIP_BOX</u>	Sets or restores clipping rectangle
<u>SETCOLORTABLE</u>	Sets RGB color-table entry
<u>SETCOPYCOUNT</u>	Superseded: Use ExtDeviceMode function
<u>SETKERNTRACK</u>	Sets kerning track
<u>SETLINECAP</u>	Sets line-end style
<u>SETLINEJOIN</u>	Sets line-intersection style
<u>SETMITERLIMIT</u>	Sets line-intersection bevel angle
<u>SET_POLY_MODE</u>	Sets mode for Polygon and Polyline functions
<u>SET_SCREEN_ANGLE</u>	Sets current screen angle
<u>SET_SPREAD</u>	Sets trapping for spot separations
<u>STARTDOC</u>	Superseded: Use StartDoc function
<u>STRETCHBLT</u>	Superseded: Use StretchBlt function
<u>TRANSFORM_CTM</u>	Modifies current transformation matrix

## ACCELERATORS (2.x)

acctablename ACCELERATORS

BEGIN

event, idvalue, [type] [options]

.

.

END

The ACCELERATORS statement defines one or more accelerators for an application. An accelerator is a keystroke defined by the application to give the user a quick way to perform a task. The TranslateAccelerator function is used to translate accelerator messages from the application queue into WM\_COMMAND or WM\_SYSCOMMAND messages.

Parameter	Description										
acctablename	Specifies either a unique name or an integer value that identifies the resource.										
event	Specifies the keystroke to be used as an accelerator. It can be any one of the following character types: <table><thead><tr><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>"char"</td><td>A single ASCII character enclosed in double quotation marks. The character can be preceded by a caret (^), meaning that the character is a control character.</td></tr><tr><td>ASCII character</td><td>An integer value representing an ASCII character. The type parameter must be ASCII.</td></tr><tr><td>Virtual-key character</td><td>An integer value representing a virtual key. The virtual key for alphanumeric keys can be specified by placing the uppercase letter or number in double quotation marks (for example, "9" or "C"). The type parameter must be VIRTKEY.</td></tr></tbody></table>	Type	Description	"char"	A single ASCII character enclosed in double quotation marks. The character can be preceded by a caret (^), meaning that the character is a control character.	ASCII character	An integer value representing an ASCII character. The type parameter must be ASCII.	Virtual-key character	An integer value representing a virtual key. The virtual key for alphanumeric keys can be specified by placing the uppercase letter or number in double quotation marks (for example, "9" or "C"). The type parameter must be VIRTKEY.		
Type	Description										
"char"	A single ASCII character enclosed in double quotation marks. The character can be preceded by a caret (^), meaning that the character is a control character.										
ASCII character	An integer value representing an ASCII character. The type parameter must be ASCII.										
Virtual-key character	An integer value representing a virtual key. The virtual key for alphanumeric keys can be specified by placing the uppercase letter or number in double quotation marks (for example, "9" or "C"). The type parameter must be VIRTKEY.										
idvalue	Specifies an integer value that identifies the accelerator.										
type	Required only when the event parameter is an ASCII character or a virtual-key character. The type parameter specifies either ASCII or VIRTKEY; the integer value of event is interpreted accordingly. When VIRTKEY is specified and event contains a string, event must be uppercase.										
options	Specifies the options that define the accelerator. This parameter can be one or more of the following values: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>NOINVERT</td><td>Specifies that no top-level menu item is highlighted when the accelerator is used. This is useful when defining accelerators for actions such as scrolling that do not correspond to a menu item. If NOINVERT is omitted, a top-level menu item will be highlighted (if possible) when the accelerator is used.</td></tr><tr><td>ALT</td><td>Causes the accelerator to be activated only if the ALT key is down.</td></tr><tr><td>SHIFT</td><td>Causes the accelerator to be activated only if the SHIFT key is down.</td></tr><tr><td><u>CONTROL</u></td><td>Defines the character as a control character (the accelerator is only activated if the <u>CONTROL</u> key is down). This has the same effect as using a caret (^) before the accelerator character in the event parameter.</td></tr></tbody></table>	Option	Description	NOINVERT	Specifies that no top-level menu item is highlighted when the accelerator is used. This is useful when defining accelerators for actions such as scrolling that do not correspond to a menu item. If NOINVERT is omitted, a top-level menu item will be highlighted (if possible) when the accelerator is used.	ALT	Causes the accelerator to be activated only if the ALT key is down.	SHIFT	Causes the accelerator to be activated only if the SHIFT key is down.	<u>CONTROL</u>	Defines the character as a control character (the accelerator is only activated if the <u>CONTROL</u> key is down). This has the same effect as using a caret (^) before the accelerator character in the event parameter.
Option	Description										
NOINVERT	Specifies that no top-level menu item is highlighted when the accelerator is used. This is useful when defining accelerators for actions such as scrolling that do not correspond to a menu item. If NOINVERT is omitted, a top-level menu item will be highlighted (if possible) when the accelerator is used.										
ALT	Causes the accelerator to be activated only if the ALT key is down.										
SHIFT	Causes the accelerator to be activated only if the SHIFT key is down.										
<u>CONTROL</u>	Defines the character as a control character (the accelerator is only activated if the <u>CONTROL</u> key is down). This has the same effect as using a caret (^) before the accelerator character in the event parameter.										

The ALT, SHIFT, and CONTROL options apply only to virtual keys.

### Example

The following example demonstrates the usage of accelerator keys:

```
1 ACCELERATORS
BEGIN
  "^C", IDDCLEAR ; control C
  "K", IDDCLEAR ; shift K
  "k", IDDELLIPSE, ALT ; alt k
```

```
98,    IDIRECT, ASCII    ; b
66,    IDSTAR, ASCII    ; B (shift b)
"g",   IDIRECT; g
"G",   IDSTAR; G (shift G)
VK_F1, IDDCLEAR, VIRTKEY ; F1
VK_F1, IDSTAR, CONTROL, VIRTKEY    ; control F1
VK_F1, IDDELLIPSE, SHIFT, VIRTKEY  ; shift F1
VK_F1, IDIRECT, ALT, VIRTKEY    ; alt F1
VK_F2, IDDCLEAR, ALT, SHIFT, VIRTKEY    ; alt shift F2
VK_F2, IDSTAR, CONTROL, SHIFT, VIRTKEY  ; ctrl shift F2
VK_F2, IDIRECT, ALT, CONTROL, VIRTKEY    ; alt control F2
END
```

See Also  
[TranslateAccelerator](#)

## BITMAP (3.0)

nameID BITMAP [load-option] [mem-option] filename

The BITMAP resource-definition statement specifies a custom bitmap that an application uses in its screen display or as an item in a menu.

Parameter	Description								
nameID	Specifies either a unique name or an integer value identifying the resource.								
load-option	Specifies when the resource is to be loaded. The parameter must be one of the following: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>PRELOAD</td><td>Resource is loaded immediately.</td></tr><tr><td>LOADONCALL</td><td>Resource is loaded when called. This is the default option.</td></tr></tbody></table>	Option	Description	PRELOAD	Resource is loaded immediately.	LOADONCALL	Resource is loaded when called. This is the default option.		
Option	Description								
PRELOAD	Resource is loaded immediately.								
LOADONCALL	Resource is loaded when called. This is the default option.								
mem-option	Specifies whether the resource is fixed or movable and whether it is discardable. The parameter must be one of the following: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td><u>FIXED</u></td><td>Resource remains at a fixed memory location.</td></tr><tr><td>MOVEABLE</td><td>Resource can be moved if necessary in order to compact memory.</td></tr><tr><td>DISCARDABLE</td><td>Resource can be discarded if no longer needed.</td></tr></tbody></table> The default for bitmap resources is MOVEABLE.	Option	Description	<u>FIXED</u>	Resource remains at a fixed memory location.	MOVEABLE	Resource can be moved if necessary in order to compact memory.	DISCARDABLE	Resource can be discarded if no longer needed.
Option	Description								
<u>FIXED</u>	Resource remains at a fixed memory location.								
MOVEABLE	Resource can be moved if necessary in order to compact memory.								
DISCARDABLE	Resource can be discarded if no longer needed.								
filename	Specifies the name of the file that contains the resource. The name must be a valid MS-DOS filename; it must be a full path if the file is not in the current working directory. The path can either be a quoted or non-quoted string.								

### Example

The following example specifies two bitmap resources:

```
disk1 BITMAP disk.bmp
12 BITMAP PRELOAD diskette.bmp
```

See Also

LoadBitmap

## CAPTION (2.x)

CAPTION captiontext

The CAPTION statement defines the title for the dialog box. The title appears in the box's caption bar (if it has one).

The default caption is empty.

Parameter	Description
captiontext	Specifies an ASCII character string enclosed in double quotation marks.

### Example

The following example demonstrates the usage of the CAPTION statement:

```
CAPTION "Error!"
```

## CHECKBOX (2.x)

CHECKBOX text, id, x, y, width, height, [style]

The CHECKBOX statement creates a check box control. The control is a small rectangle (check box) that has the specified text displayed next to it (typically, to the right). When the user selects the control, the control highlights the rectangle and sends a message to its parent window. The CHECKBOX statement, which can only be used in a DIALOG statement, defines the text, identifier, dimensions, and attributes of the control.

Parameter	Description
text	Specifies text that is displayed to the right of the control. This parameter must contain zero or more characters enclosed in double quotation marks. Character values must be in the range 1 through 255. If a double quotation mark is required in the text, you must include the double quotation mark twice. An ampersand (&) character in the text indicates that the following character is used as a mnemonic character for the control. When the control is displayed, the ampersand is not shown, but the mnemonic character is underlined. The user can choose the control by pressing the key corresponding to the underlined mnemonic character. To use the ampersand as a character in a string, insert two ampersands (&&).
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or a simple expression that evaluates to a value in that range.
x	Specifies the x-coordinate of the left side of the control relative to the left side of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (-) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
y	Specifies the y-coordinate of the top side of the control relative to the top of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (-) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
width	Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (-) operator. The width is in 1/4-character units.
height	Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (-) operator. The height is in 1/8-character units.
style	Specifies the control styles. This value can be a combination of the button class style <u>BS_CHECKBOX</u> and the <u>WS_TABSTOP</u> and <u>WS_GROUP</u> styles. You can use the bitwise OR (!) operator to combine styles. If you do not specify a style, the default style is <u>BS_CHECKBOX</u> and <u>WS_TABSTOP</u> .

### Comments

The current dialog units are computed from the height and width of the current system font. The GetDialogBaseUnits function returns the dialog units in pixels.

### Example

This example creates a check-box control that is labeled "Italic":

```
CHECKBOX "Italic", 3, 10, 10, 40, 10
```

### See Also

GetDialogBaseUnits



## CLASS (2.x)

### CLASS class

The CLASS statement defines the class of the dialog box. If no statement is given, the Windows standard dialog class will be used as the default.

Parameter	Description
class	Specifies an integer or a string, enclosed in double quotation marks, that identifies the class of the dialog box. If the window procedure for the class does not process a message sent to it, it must call the <u>DefDlgProc</u> function to ensure that all messages are handled properly for the dialog box. A <u>private</u> class can use DefDlgProc as the default window procedure. The class must be registered with the cbWndExtra member of the <u>WNDCLASS</u> structure set to DLGWINDOWEXTRA.

### Comments

The CLASS statement should only be used with special cases, since it overrides the normal processing of a dialog box. The CLASS statement converts a dialog box to a window of the specified class; depending on the class, this could give undesirable results. Do not use the predefined control-class names with this statement.

### Example

The following example demonstrates the usage of the CLASS statement:

```
CLASS "myclass"
```

### See Also

DefDlgProc, WNDCLASS

## COMBOBOX (2.x)

COMBOBOX id, x, y, width, height[, style]

The COMBOBOX statement creates a combination box control (a combo box). A combo box consists of either a static text box or an edit box combined with a list box. The list box can be displayed at all times or pulled down by the user. If the combo box contains a static text box, the text box always displays the selection (if any) in the list box portion of the combo box. If it uses an edit box, the user can type in the desired selection; the list box highlights the first item (if any) that matches what the user has entered in the edit box. The user can then select the item highlighted in the list box to complete the choice. In addition, the combo box can be owner-drawn and of fixed or variable height.

Parameter	Description
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or a simple expression that evaluates to a value in that range.
x	Specifies the x-coordinate of the left side of the control relative to the left side of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
y	Specifies the y-coordinate of the top side of the control relative to the top of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
width	Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The width is in 1/4-character units.
height	Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The height is in 1/8-character units. This value specifies the entire height of the control, regardless of whether the entire control is initially displayed.
style	Specifies the control styles. This value can be a combination of the COMBOBOX class styles (see the Combination-box styles topic) and any of the following styles: <u>WS_TABSTOP</u> , <u>WS_GROUP</u> , <u>WS_VSCROLL</u> , and <u>WS_DISABLED</u> . You can use the bitwise OR ( ) operator to combine styles. If you do not specify a style, the default style is <u>CBS_SIMPLE</u> and <u>WS_TABSTOP</u> .

### Comments

The current dialog units are computed from the height and width of the current system font. The GetDialogBaseUnits function returns the dialog units in pixels.

### Example

This example creates a combo-box control with a vertical scroll bar:

```
COMBOBOX 777, 10, 10, 50, 54, CBS_SIMPLE | WS_VSCROLL | WS_TABSTOP
```

## CONTROL (2.x)

CONTROL text, id, class, style, x, y, width, height

The CONTROL statement defines a control as belonging to the specified class. The statement defines the position and dimensions of the control within the parent window as well as the control style. The CONTROL statement is most often used in a DIALOG statement.

Parameter	Description
text	Specifies displayed text. Its position depends on the control class. This parameter must contain zero or more characters enclosed in double quotation marks. Character values must be in the range 1 through 255. If a double quotation mark is required in the text, you must include the double quotation mark twice. In the appropriate styles, an ampersand (&) character in the text indicates that the following character is used as a mnemonic character for the control. When the control is displayed, the ampersand is not shown, but the mnemonic character is underlined. The user can choose the control by pressing the key corresponding to the character.
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or a simple expression that evaluates to a value in that range.
class	Specifies the control class. This value can be a predefined name, character string, or integer value that defines the class. This value can be one of the classes specified in the topic <u>Control Classes</u> .
style	Specifies the control style. For a list of possible control styles, see the topic <u>Control Styles</u> . You can use the bitwise OR (!) operator to combine styles.
x	Specifies the x-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the parent window.
y	Specifies the y-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the parent window.
width	Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The value is in 1/4-character units.
height	Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The value is in 1/8-character units.

## CTEXT (2.x)

CTEXT text, id, x, y, width, height[, style]

The CTEXT statement creates a centered-text control. The control is a simple rectangle displaying the given text centered in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next line. The CTEXT statement, which you can use only in a DIALOG statement, defines the text, identifier, dimensions, and attributes of the control.

Parameter	Description
text	Specifies text that is centered in the rectangular area of the control. This parameter must contain zero or more characters enclosed in double quotation marks. Character values must be in the range 1 through 255. If a double quotation mark is required in the text, you must include the double quotation mark twice.
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or a simple expression that evaluates to a value in that range.
x	Specifies the x-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
y	Specifies the y-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
width	Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The width is in 1/4-character units.
height	Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The height is in 1/8-character units.
style	Specifies the control styles. This value can be any combination of the following styles: <u>SS_CENTER</u> , <u>WS_TABSTOP</u> , and <u>WS_GROUP</u> . You can use the bitwise OR (!) operator to combine styles. If you do not specify a style, the default style is <u>SS_CENTER</u> and <u>WS_GROUP</u> .

### Example

This example creates a centered-text control that is labeled "Filename":

```
CTEXT "Filename", 101, 10, 10, 100, 100
```

### See Also

CONTROL, DIALOG, LTEXT, RTEXT

## CURSOR (3.0)

nameID CURSOR [load-option] [mem-option] filename

The CURSOR statement specifies a bitmap that defines the shape of the cursor on the display screen.

Parameter	Description								
nameID	Specifies either a unique name or an integer identifying the resource.								
load-option	Specifies when the resource is to be loaded. The parameter must be one of the following: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>PRELOAD</td><td>Resource is loaded immediately.</td></tr><tr><td>LOADONCALL</td><td>Resource is loaded when called. This is the default option.</td></tr></tbody></table>	Option	Description	PRELOAD	Resource is loaded immediately.	LOADONCALL	Resource is loaded when called. This is the default option.		
Option	Description								
PRELOAD	Resource is loaded immediately.								
LOADONCALL	Resource is loaded when called. This is the default option.								
mem-option	Specifies whether the resource is fixed or movable and whether it is discardable. The parameter must be one of the following: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td><u>FIXED</u></td><td>Resource remains at a fixed memory location.</td></tr><tr><td>MOVEABLE</td><td>Resource can be moved if necessary in order to compact memory.</td></tr><tr><td>DISCARDABLE</td><td>Resource can be discarded if no longer needed.</td></tr></tbody></table> The default is MOVEABLE and DISCARDABLE for cursor, icon, and font resources. The default for bitmap resources is MOVEABLE.	Option	Description	<u>FIXED</u>	Resource remains at a fixed memory location.	MOVEABLE	Resource can be moved if necessary in order to compact memory.	DISCARDABLE	Resource can be discarded if no longer needed.
Option	Description								
<u>FIXED</u>	Resource remains at a fixed memory location.								
MOVEABLE	Resource can be moved if necessary in order to compact memory.								
DISCARDABLE	Resource can be discarded if no longer needed.								
filename	Specifies the name of the file that contains the resource. The name must be a valid MS-DOS filename; it must be a full path if the file is not in the current working directory. The path can either be a quoted or non-quoted string.								

### Comments

Icon and cursor resources can contain more than one image. If the resource is marked with the PRELOAD option, Windows loads all images in the resource when the application executes.

### Example

The following example specifies two cursor resources; one by name (cursor1) and the other by number (2):

```
cursor1 CURSOR bullseye.cur
2 CURSOR "d:\\cursor\\arrow.cur"
```

## #define (2.x)

#define name value

The #define directive assigns the given value to the specified name. All subsequent occurrences of the name are replaced by the value.

Parameter	Description
name	Specifies the name to be defined. This value is any combination of letters, digits, and punctuation.
value	Specifies any integer, character string, or line of text.

### Example

This example assigns values to the names "NONZERO" and "USERCLASS":

```
#defineNONZERO1  
#defineUSERCLASS "MyControlClass"
```

### See Also

#ifdef, #ifndef, #undef

## DEFPUSHBUTTON (2.x)

DEFPUSHBUTTON text, id, x, y, width, height[, style]

The DEFPUSHBUTTON statement creates a default push-button control. The control is a small rectangle with a bold outline that represents the default response for the user. The given text is displayed inside the button. The control highlights the button in the usual way when the user clicks the mouse in it and sends a message to its parent window.

Parameter	Description
text	Specifies text that is centered in the rectangular area of the control. This parameter must contain zero or more characters enclosed in double quotation marks. Character values must be in the range 1 through 255. If a double quotation mark is required in the text, you must include the double quotation mark twice. An ampersand (&) character in the text indicates that the following character is used as a mnemonic character for the control. When the control is displayed, the ampersand is not shown but the mnemonic character is underlined. The user can choose the control by pressing the key corresponding to the underlined mnemonic character. To use the ampersand as a character in a string, insert two ampersands (&&).
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or a simple expression that evaluates to a value in that range.
x	Specifies the x-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (-) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
y	Specifies the y-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (-) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
width	Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (-) operator. The width is in 1/4-character units.
height	Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (-) operator. The height is in 1/8-character units.
style	Specifies the control styles. This value can be a combination of the following styles: <u>BS_DEFPUSHBUTTON</u> , <u>WS_TABSTOP</u> , <u>WS_GROUP</u> , and <u>WS_DISABLED</u> . You can use the bitwise OR ( ) operator to combine styles. If you do not specify a style, the default style is <u>BS_DEFPUSHBUTTON</u> and <u>WS_TABSTOP</u> .

### Example

This example creates a default push-button control that is labeled "Cancel":

```
DEFPUSHBUTTON "Cancel", 101, 10, 10, 24, 50
```

See Also

PUSHBUTTON, RADIOBUTTON

## DIALOG (2.x)

nameID DIALOG [load-option] [mem-option] x, y, width, height

BEGIN  
control-statements

·  
·  
·

END

The DIALOG statement defines a window that an application can use to create dialog boxes. The statement defines the position and dimensions of the dialog box on the screen as well as the dialog box style.

Parameter	Description								
nameID	Identifies the dialog box. This is either a unique name or a unique integer value in the range 1 to 65,535.								
load-option	Specifies when the resource is to be loaded. This parameter is optional. If it is specified, it must be one of the following: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>PRELOAD</td><td>Resource is loaded immediately.</td></tr><tr><td>LOADONCALL</td><td>Resource is loaded when called. This is the default option.</td></tr></tbody></table>	Option	Description	PRELOAD	Resource is loaded immediately.	LOADONCALL	Resource is loaded when called. This is the default option.		
Option	Description								
PRELOAD	Resource is loaded immediately.								
LOADONCALL	Resource is loaded when called. This is the default option.								
mem-option	Specifies whether the resource is fixed or movable and whether it is discardable. This parameter is optional. If it is specified, it must be either <b>FIXED</b> or <b>MOVEABLE</b> . An additional value, <b>DISCARDABLE</b> may also be specified. The following list describes the options in more detail: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td><b>FIXED</b></td><td>Resource remains at a fixed memory location.</td></tr><tr><td><b>MOVEABLE</b></td><td>Resource can be moved if necessary in order to compact memory. This is the default option.</td></tr><tr><td><b>DISCARDABLE</b></td><td>Resource can be discarded if no longer needed.</td></tr></tbody></table>	Option	Description	<b>FIXED</b>	Resource remains at a fixed memory location.	<b>MOVEABLE</b>	Resource can be moved if necessary in order to compact memory. This is the default option.	<b>DISCARDABLE</b>	Resource can be discarded if no longer needed.
Option	Description								
<b>FIXED</b>	Resource remains at a fixed memory location.								
<b>MOVEABLE</b>	Resource can be moved if necessary in order to compact memory. This is the default option.								
<b>DISCARDABLE</b>	Resource can be discarded if no longer needed.								
x	Specifies the x-coordinate of the left side of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units.								
y	Specifies the y-coordinate of the top side of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units.								
width	Specifies the width of the dialog box. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The width is in 1/4-character units.								
height	Specifies the height of the dialog box. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The height is in 1/8-character units.								
style	Specifies the dialog box styles. This parameter can be one of the following values: <table><thead><tr><th>Style</th><th>Meaning</th></tr></thead><tbody><tr><td><b>DS_LOCALEDIT</b></td><td>Specifies that edit controls in the dialog box will use memory in the application's data segment. By default, all edit controls in dialog boxes use memory outside the application's data segment. This feature may be suppressed by adding the <b>DS_LOCALEDIT</b> flag to the Style command for the dialog box. If this flag is not used, <b>EM_GETHANDLE</b> and <b>EM_SETHANDLE</b> messages must not be used, because the storage for the control is not in the application's data segment. This feature does not affect edit controls created outside of dialog boxes.</td></tr><tr><td><b>DS_MODALFRAME</b></td><td>Creates a dialog box with a modal dialog box frame that can be combined with a title bar and System menu by specifying the <b>WS_CAPTION</b> and <b>WS_SYSMENU</b> styles.</td></tr></tbody></table>	Style	Meaning	<b>DS_LOCALEDIT</b>	Specifies that edit controls in the dialog box will use memory in the application's data segment. By default, all edit controls in dialog boxes use memory outside the application's data segment. This feature may be suppressed by adding the <b>DS_LOCALEDIT</b> flag to the Style command for the dialog box. If this flag is not used, <b>EM_GETHANDLE</b> and <b>EM_SETHANDLE</b> messages must not be used, because the storage for the control is not in the application's data segment. This feature does not affect edit controls created outside of dialog boxes.	<b>DS_MODALFRAME</b>	Creates a dialog box with a modal dialog box frame that can be combined with a title bar and System menu by specifying the <b>WS_CAPTION</b> and <b>WS_SYSMENU</b> styles.		
Style	Meaning								
<b>DS_LOCALEDIT</b>	Specifies that edit controls in the dialog box will use memory in the application's data segment. By default, all edit controls in dialog boxes use memory outside the application's data segment. This feature may be suppressed by adding the <b>DS_LOCALEDIT</b> flag to the Style command for the dialog box. If this flag is not used, <b>EM_GETHANDLE</b> and <b>EM_SETHANDLE</b> messages must not be used, because the storage for the control is not in the application's data segment. This feature does not affect edit controls created outside of dialog boxes.								
<b>DS_MODALFRAME</b>	Creates a dialog box with a modal dialog box frame that can be combined with a title bar and System menu by specifying the <b>WS_CAPTION</b> and <b>WS_SYSMENU</b> styles.								



DS_NOIDLEMSG	Suppresses <u>WM_ENTERIDLE</u> messages that Windows would otherwise send to the owner of the dialog box while the dialog box is displayed.
DS_SYSMODAL	Creates a system-modal dialog box.

#### Comments

The GetDialogBaseUnits function returns the dialog base units in pixels. The exact meaning of the coordinates depends on the style defined by the STYLE option statement. For child-style dialog boxes, the coordinates are relative to the origin of the parent window, unless the dialog box has the style DS\_ABSALIGN; in that case, the coordinates are relative to the origin of the display screen.

Do not use the WS\_CHILD style with a modal dialog box. The DialogBox function always disables the parent/owner of the newly created dialog box. When a parent window is disabled, its child windows are implicitly disabled. Since the parent window of the child-style dialog box is disabled, the child-style dialog box is too.

If a dialog box has the DS\_ABSALIGN style, the dialog coordinates for its upper-left corner are relative to the screen origin instead of to the upper-left corner of the parent window. You would typically use this style when you wanted the dialog box to start in a specific part of the display no matter where the parent window may be on the screen.

The name DIALOG can also be used as the class-name parameter to the CreateWindow function to create a window with dialog box attributes.

#### Example

The following demonstrates the usage of the DIALOG statement:

```
#include <windows.h>

ErrorDialog DIALOG 10, 10, 300, 110
STYLE WS_POPUP|WS_BORDER
CAPTION "Error!"
BEGIN
    CTEXT "Select One:", 1, 10, 10, 280, 12
    PUSHBUTTON "&Retry", 2, 75, 30, 60, 12
    PUSHBUTTON "&Abort", 3, 75, 50, 60, 12
    PUSHBUTTON "&Ignore", 4, 75, 80, 60, 12
END
```

#### See Also

CreateDialog, CreateWindow, DialogBox, GetDialogBaseUnits

## EDITTEXT (2.x)

EDITTEXT id, x, y, width, height[, style]

The EDITTEXT statement defines an EDIT control belonging to the EDIT class. It creates a rectangular region in which the user can enter and edit text. The control displays a cursor when the user clicks the mouse in it. The user can then use the keyboard to enter text or edit the existing text. Editing keys include the BACKSPACE and DELETE keys. The user can also use the mouse to select characters to be deleted or to select the place to insert new characters.

Parameter	Description
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or a simple expression that evaluates to a value in that range.
x	Specifies the x-coordinate of the left side of the control relative to the left side of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
y	Specifies the y-coordinate of the top side of the control relative to the top of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
width	Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The width is in 1/4-character units.
height	Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The height is in 1/8-character units.
style	Specifies the control styles. This value can be a combination of the edit class styles (see the <a href="#">Edit-control styles</a> topic) and the following styles: <a href="#">WS_TABSTOP</a> , <a href="#">WS_GROUP</a> , <a href="#">WS_VSCROLL</a> , <a href="#">WS_HSCROLL</a> , and <a href="#">WS_DISABLED</a> . You can use the bitwise OR ( ) operator to combine styles. If you do not specify a style, the default style is <a href="#">ES_LEFT</a> , <a href="#">WS_BORDER</a> , and <a href="#">WS_TABSTOP</a> .

### Example

The following example demonstrates the usage of the EDITTEXT statement:

```
EDITTEXT 3, 10, 10, 100, 10
```

## #elif (2.x)

### #elif constant-expression

The #elif directive marks an optional clause of a conditional-compilation block defined by a #ifdef, #ifndef, or #if directive. The directive controls conditional compilation of the resource file by checking the specified constant expression. If the constant expression is nonzero, #elif directs the compiler to continue processing statements up to the next #endif, #else, or #elif directive and then skip to the statement after #endif. If the constant expression is zero, #elif directs the compiler to skip to the next #endif, #else, or #elif directive. You can use any number of #elif directives in a conditional block.

Parameter	Description
constant-expression	Specifies the expression to be checked. This value is a defined name, an integer constant, or an expression consisting of names, integers, and arithmetic and relational operators.

### Example

In this example, #elif directs the compiler to process the second BITMAP statement only if the value assigned to the name "Version" is less than 7. The #elif directive itself is processed only if Version is greater than or equal to 3.

```
#if Version < 3
BITMAP 1 errbox.bmp
#elif Version < 7
BITMAP 1 userbox.bmp
#endif
```

### See Also

#else, #endif, #if, #ifdef, #ifndef

## #else (2.x)

### #else

The #else directive marks an optional clause of a conditional-compilation block defined by a #ifdef, #ifndef, or #if directive. The #else directive must be the last directive before the #endif directive.

This directive has no arguments.

### Example

This example compiles the second BITMAP statement only if the name "DEBUG" is not defined:

```
#ifdef DEBUG
    BITMAP 1 errbox.bmp
#else
    BITMAP 1 userbox.bmp
#endif
```

### See Also

#elif, #endif, #if, #ifdef, #ifndef

## #endif (2.x)

### #endif

The #endif directive marks the end of a conditional-compilation block defined by a #ifdef directive. One #endif is required for each #if, #ifdef, or #ifndef directive.

This directive has no arguments.

### See Also

#elif, #else, #if, #ifdef, #ifndef

## FONT (2.x)

FONT pointsize, typeface

The FONT statement defines the font with which Windows will draw text in the dialog box. The font must have been previously loaded, either from the WIN.INI file or by calling the LoadResource function.

Parameter	Description
pointsize	Specifies the size, in points, of the font.
typeface	Specifies the name of the typeface. This name must be identical to the name defined in the [fonts] section of WIN.INI. This parameter must be enclosed in double quotes.

### Example

The following example demonstrates the usage of the FONT statement:

```
FONT 12, "MS Sans Serif"
```

See Also

DIALOG, LoadResource

## FONT (3.0)

nameID FONT [load-option] [mem-option] filename

The FONT resource-definition statement specifies a file that contains a font.

For a font resource, nameID must be a number; it cannot be a name.

Parameter	Description								
nameID	Specifies either a unique name or an integer value identifying the resource.								
load-option	Specifies when the resource is to be loaded. The parameter must be one of the following options: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>PRELOAD</td><td>Resource is loaded immediately.</td></tr><tr><td>LOADONCALL</td><td>Resource is loaded when called. This is the default option.</td></tr></tbody></table>	Option	Description	PRELOAD	Resource is loaded immediately.	LOADONCALL	Resource is loaded when called. This is the default option.		
Option	Description								
PRELOAD	Resource is loaded immediately.								
LOADONCALL	Resource is loaded when called. This is the default option.								
mem-option	Specifies whether the resource is fixed or movable and whether it is discardable. The parameter must be one of the following options: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>FIXED</td><td>Resource remains at a fixed memory location.</td></tr><tr><td>MOVEABLE</td><td>Resource can be moved if necessary in order to compact memory.</td></tr><tr><td>DISCARDABLE</td><td>Resource can be discarded if no longer needed.</td></tr></tbody></table> The default is MOVEABLE and DISCARDABLE for cursor, icon, and font resources. The default for bitmap resources is MOVEABLE.	Option	Description	FIXED	Resource remains at a fixed memory location.	MOVEABLE	Resource can be moved if necessary in order to compact memory.	DISCARDABLE	Resource can be discarded if no longer needed.
Option	Description								
FIXED	Resource remains at a fixed memory location.								
MOVEABLE	Resource can be moved if necessary in order to compact memory.								
DISCARDABLE	Resource can be discarded if no longer needed.								
filename	Specifies the name of the file that contains the resource. The name must be a valid MS-DOS filename; it must be a full path if the file is not in the current working directory. The path can either be a quoted or non-quoted string.								

### Example

The following example specifies a single font resource:

```
5 FONT CMROMAN.FNT
```

## GROUPBOX (2.x)

GROUPBOX text, id, x, y, width, height[, style]

The GROUPBOX statement creates a group box control. The control is a rectangle that groups other controls together. The controls are grouped by drawing a border around them and displaying the given text in the upper-left corner. The GROUPBOX statement, which you can use only in a DIALOG statement, defines the text, identifier, dimensions, and attributes of a control window.

Parameter	Description
text	Specifies text that is displayed to the right of the control. This parameter must contain zero or more characters enclosed in double quotation marks. Character values must be in the range 1 through 255. If a double quotation mark is required in the text, you must include the double quotation mark twice. An ampersand (&) character in the text indicates that the following character is used as a mnemonic character for the control. When the control is displayed, the ampersand is not shown but the mnemonic character is underlined. The user can choose the control by pressing the key corresponding to the underlined mnemonic character. To use the ampersand as a character in a string, insert two ampersands (&&).
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or a simple expression that evaluates to a value in that range.
x	Specifies the x-coordinate of the left side of the control relative to the left side of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
y	Specifies the y-coordinate of the top side of the control relative to the top of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
width	Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The width is in 1/4-character units.
height	Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The height is in 1/8-character units.
style	Specifies the control styles. This value can be a combination of the button class style <u>BS_GROUPBOX</u> and the <u>WS_TABSTOP</u> and <u>WS_DISABLED</u> styles. You can use the bitwise OR ( ) operator to combine styles. If you do not specify a style, the default style is <u>BS_GROUPBOX</u> .

### Example

This example creates a group-box control that is labeled "Options":

```
GROUPBOX "Options", 101, 10, 10, 100, 100
```

See Also  
DIALOG



## ICON (2.x)

ICON text, id, x, y, [width, height, style]

The ICON statement creates an icon control. This control is an icon displayed in a dialog box. The ICON statement, which you can use only in a DIALOG statement, defines the icon-resource identifier, icon-control identifier, position, and attributes of a control.

Parameter	Description
text	Specifies the name of an icon (not a filename) defined elsewhere in the resource file.
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or a simple expression that evaluates to a value in that range.
x	Specifies the x-coordinate of the left side of the control relative to the left side of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
y	Specifies the y-coordinate of the top side of the control relative to the top of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
width	This value is ignored and should be set to zero.
height	This value is ignored and should be set to zero.
style	Specifies the control style. This parameter is optional. The only value that can be specified is the <u>SS_ICON</u> style. This is the default style whether this parameter is specified or not.

### Example

This example creates an icon control whose icon identifier is 901 and whose name is "myicon":

```
ICON "myicon" 901, 30, 30
```

See Also

DIALOG

## ICON (3.0)

nameID ICON [load-option] [mem-option] filename

The ICON resource-definition statement specifies a bitmap that defines the shape of the icon to be used for a given application.

Parameter	Description								
nameID	Specifies either a unique name or an integer value identifying the resource.								
load-option	Specifies when the resource is to be loaded. The parameter must be one of the following options: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>PRELOAD</td><td>Resource is loaded immediately.</td></tr><tr><td>LOADONCALL</td><td>Resource is loaded when called. This is the default option.</td></tr></tbody></table>	Option	Description	PRELOAD	Resource is loaded immediately.	LOADONCALL	Resource is loaded when called. This is the default option.		
Option	Description								
PRELOAD	Resource is loaded immediately.								
LOADONCALL	Resource is loaded when called. This is the default option.								
mem-option	Specifies whether the resource is fixed or movable and whether it is discardable. The parameter must be one of the following options: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>FIXED</td><td>Resource remains at a fixed memory location.</td></tr><tr><td>MOVEABLE</td><td>Resource can be moved if necessary in order to compact memory.</td></tr><tr><td>DISCARDABLE</td><td>Resource can be discarded if no longer needed.</td></tr></tbody></table> The default is MOVEABLE and DISCARDABLE for cursor, icon, and font resources. The default for bitmap resources is MOVEABLE.	Option	Description	FIXED	Resource remains at a fixed memory location.	MOVEABLE	Resource can be moved if necessary in order to compact memory.	DISCARDABLE	Resource can be discarded if no longer needed.
Option	Description								
FIXED	Resource remains at a fixed memory location.								
MOVEABLE	Resource can be moved if necessary in order to compact memory.								
DISCARDABLE	Resource can be discarded if no longer needed.								
filename	Specifies the name of the file that contains the resource. The name must be a valid MS-DOS filename; it must be a full path if the file is not in the current working directory. The path can either be a quoted or non-quoted string.								

### Comments

Icon and cursor resources can contain more than one image. If the resource is marked as PRELOAD, Windows loads all images in the resource when the application executes.

### Example

The following example specifies two icon resources:

```
desk1  ICON desk.ico
11 ICON DISCARDABLE custom.ico
```

## #if (2.x)

### #if constant-expression

The #if directive controls conditional compilation of the resource file by checking the specified constant expression. If the constant expression is nonzero, #if directs the compiler to continue processing statements up to the next #endif, #else, or #elif directive and then skip to the statement after the #endif directive. If the constant expression is zero, #if directs the compiler to skip to the next #endif, #else, or #elif directive.

Parameter	Description
constant-expression	Specifies the expression to be checked. This value is a defined name, an integer constant, or an expression consisting of names, integers, and arithmetic and relational operators.

### Example

This example compiles the BITMAP statement only if the value assigned to the name "Version" is less than 3:

```
#if Version < 3
BITMAP 1 errbox.bmp
#endif
```

### See Also

#elif, #else, #endif, #ifdef, #ifndef

## #ifdef (2.x)

#ifdef name

The #ifdef directive controls conditional compilation of the resource file by checking the specified name. If the name has been defined by using a #define directive or by using the -d command-line option with the Resource Compiler, #ifdef directs the compiler to continue with the statement immediately after the #ifdef directive. If the name has not been defined, #ifdef directs the compiler to skip all statements up to the next #endif directive.

Parameter	Description
name	Specifies the name to be checked by the directive.

Example

This example compiles the BITMAP statement only if the name "Debug" is defined:

```
#ifdef Debug
BITMAP 1 errbox.bmp
#endif
```

See Also

#define, #endif, #if, #ifndef, #undef

## #ifndef (2.x)

#ifndef name

The #ifndef directive controls conditional compilation of the resource file by checking the specified name. If the name has not been defined or if its definition has been removed by using the #undef directive, #ifndef directs the compiler to continue processing statements up to the next #endif, #else, or #elif directive and then skip to the statement after the #endif directive. If the name is defined, #ifndef directs the compiler to skip to the next #endif, #else, or #elif directive.

Parameter	Description
name	Specifies the name to be checked by the directive.

### Example

This example compiles the BITMAP statement only if the name "Optimize" is not defined:

```
#ifndef Optimize
BITMAP 1 errbox.bmp
#endif
```

### See Also

#elif, #else, #endif, #if, #ifdef, #undef

## #include (2.x)

### #include (filename)

The #include directive causes Resource Compiler to process the file specified in the filename parameter. This file should be a header file that defines the constants used in the resource-definition file.

Parameter	Description
filename	Specifies the name of the file to be included. This value must be an ASCII string. If the file is in the current directory, the string must be enclosed in double quotation marks; if the file is in the directory specified by the INCLUDE environment variable, the string must be enclosed in less-than and greater-than characters (<>). You must give a full path enclosed in double quotation marks if the file is not in the current directory or in the directory specified by the INCLUDE environment variable.

### Example

This example processes the header files WINDOWS.H and HEADERS\MYDEFS.H while compiling the resource-definition file:

```
#include <windows.h>
#include "headers\mydefs.h"
```

### See Also

#define

## LISTBOX (2.x)

LISTBOX id, x, y, width, height[, style]

The LISTBOX statement creates commonly used controls for a dialog box or window. The control is a rectangle containing a list of strings (such as filenames) from which the user can select. The LISTBOX statement, which can only be used in a DIALOG or WINDOW statement, defines the identifier, dimensions, and attributes of a control window.

Parameter	Description
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or a simple expression that evaluates to a value in that range.
x	Specifies the x-coordinate of the left side of the control relative to the left side of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
y	Specifies the y-coordinate of the top side of the control relative to the top of the dialog box. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
width	Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The width is in 1/4-character units.
height	Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The height is in 1/8-character units.
style	Specifies the control styles. This value can be a combination of the list-box class styles (see the <u>List-box styles</u> topic) and any of the following styles: <u>WS_BORDER</u> and <u>WS_VSCROLL</u> . You can use the bitwise OR (!) operator to combine styles. If you do not specify a style, the default style is <u>LBS_NOTIFY</u> and <u>WS_BORDER</u> .

### Example

This example creates a list-box control whose identifier is 101:

```
LISTBOX 101, 10, 10, 100, 100
```

See Also

COMBOBOX, DIALOG

## LTEXT (2.x)

LTEXT text, id, x, y, width, height, [style]

The LTEXT statement creates a left-aligned text control. The control is a simple rectangle displaying the given text left-aligned in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next line. The LTEXT statement, which can be used only in a DIALOG statement, defines the text, identifier, dimensions, and attributes of the control.

Parameter	Description
text	Specifies text that is left-aligned in the rectangular area of the control. This parameter must contain zero or more characters enclosed in double quotation marks. Character values must be in the range 1 through 255. If a double quotation mark is required in the text, you must include the double quotation mark twice.
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or a simple expression that evaluates to a value in that range.
x	Specifies the x-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
y	Specifies the y-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box, window, or control containing the specified control.
width	Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The width is in 1/4-character units.
height	Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) or subtraction (–) operator. The height is in 1/8-character units.
style	Specifies the control styles. This value can be any combination of the <u>BS_RADIOBUTTON</u> style and the following styles: <u>SS_LEFT</u> , <u>WS_TABSTOP</u> , and <u>WS_GROUP</u> . You can use the bitwise OR (!) operator to combine styles. If you do not specify a style, the default style is <u>SS_LEFT</u> and <u>WS_GROUP</u> .

### Example

This example creates a left-aligned text control that is labeled "Filename":

```
LTEXT "Filename", 101, 10, 10, 100, 100
```

See Also

CONTROL, DIALOG, CTEXT, RTEXT



## MENU (2.x)

### MENU menuname

The MENU statement defines the dialog box's menu. If no statement is given, the dialog box has no menu.

Parameter	Description
menuname	Specifies the menu to use. This value is either the name of the menu or the integer identifier of the menu.

### Example

The following example demonstrates the usage of the MENU dialog statement:

```
MENU errmenu
```

See Also

MENU statement

## MENU (2.x)

menuID MENU [load-option] [mem-option]

```
BEGIN
  item-definitions
  :
  :
  :
END
```

The MENU statement defines the contents of a menu resource. A menu resource is a collection of information that defines the appearance and function of an application menu. A menu is a special input tool that lets a user select commands from a list of command names.

Parameter	Description								
menuID	Identifies the menu. This value is either a unique string or a unique integer value in the range of 1 to 65,535.								
load-option	Specifies when the resource is to be loaded. This parameter is optional. If it is specified, it must be one of the following: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>PRELOAD</td><td>Resource is loaded immediately.</td></tr><tr><td>LOADONCALL</td><td>Resource is loaded when called. This is the default option.</td></tr></tbody></table>	Option	Description	PRELOAD	Resource is loaded immediately.	LOADONCALL	Resource is loaded when called. This is the default option.		
Option	Description								
PRELOAD	Resource is loaded immediately.								
LOADONCALL	Resource is loaded when called. This is the default option.								
mem-option	Specifies whether the resource is fixed or movable and whether it is discardable. This parameter is optional. If it is specified, it must be either <b>FIXED</b> or <b>MOVEABLE</b> . An additional value, <b>DISCARDABLE</b> , may also be specified. A description of the memory options follows: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td><b>FIXED</b></td><td>Resource remains at a fixed memory location.</td></tr><tr><td><b>MOVEABLE</b></td><td>Resource can be moved if necessary in order to compact memory. This is the default option.</td></tr><tr><td><b>DISCARDABLE</b></td><td>Resource can be discarded if no longer needed.</td></tr></tbody></table> The default is <b>MOVEABLE</b> and <b>DISCARDABLE</b> .	Option	Description	<b>FIXED</b>	Resource remains at a fixed memory location.	<b>MOVEABLE</b>	Resource can be moved if necessary in order to compact memory. This is the default option.	<b>DISCARDABLE</b>	Resource can be discarded if no longer needed.
Option	Description								
<b>FIXED</b>	Resource remains at a fixed memory location.								
<b>MOVEABLE</b>	Resource can be moved if necessary in order to compact memory. This is the default option.								
<b>DISCARDABLE</b>	Resource can be discarded if no longer needed.								

### Example

Following is an example of a complete MENU statement:

```
sample MENU
BEGIN
MENUITEM "&Soup", 100
MENUITEM "S&alad", 101
POPUP "&Entree"
BEGIN
MENUITEM "&Fish", 200
MENUITEM "&Chicken", 201, CHECKED
POPUP "&Beef"
BEGIN
MENUITEM "&Steak", 301
MENUITEM "&Prime Rib", 302
END
END
MENUITEM "&Dessert", 103
END
```

### See Also

MENUITEM, POPUP, MENU dialog statement

## MENUITEM (2.x)

MENUITEM text, result, [optionlist]

The MENUITEM statement, which is optional, defines a menu item.

Parameter	Description														
text	<p>Specifies the name of the menu item. This parameter takes an ASCII string, enclosed in double quotation marks.</p> <p>The string can contain the escape characters \t and \a. The \t character inserts a tab in the string and is used to align text in columns. Tab characters should be used only in pop-up menus, not in menu bars. (For information on pop-up menus, see the <u>POPUP</u> statement.) The \a character aligns all text that follows it flush right to the <u>menu</u> bar or pop-up menu.</p> <p>To insert a double quotation mark in the string, use two double quotation marks.</p> <p>To add a mnemonic to the text string, place the ampersand (&amp;) ahead of the letter that will be the mnemonic. This will cause the letter to appear underlined in the control and to function as the mnemonic. To use the ampersand as a character in a string, insert two ampersands (&amp;&amp;).</p>														
result	<p>Specifies the result generated when the user selects the menu item. This parameter takes an integer value. Menu-item results are always integers; when the user clicks the menu-item name, the result is sent to the window that owns the menu.</p>														
optionlist	<p>Specifies the appearance of the menu item. This optional parameter takes one or more predefined menu options, separated by commas or spaces. The menu options are as follows:</p> <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>CHECKED</td><td>Item has a check mark next to it.</td></tr><tr><td>GRAYED</td><td>Item name is initially inactive and appears on the menu in gray or a lightened shade of the menu-text color.</td></tr><tr><td>HELP</td><td>Identifies a help item.</td></tr><tr><td>INACTIVE</td><td>Item name is displayed but it cannot be selected.</td></tr><tr><td>MENUBARBREAK</td><td>Same as <u>MF_MENUBREAK</u> except that for pop-up menus, it separates the new column from the old column with a vertical line.</td></tr><tr><td>MENUBREAK</td><td>Places the menu item on a new line for static menu-bar items. For pop-up menus, it places the menu item in a new column with no dividing line between the columns.</td></tr></tbody></table>	Option	Description	CHECKED	Item has a check mark next to it.	GRAYED	Item name is initially inactive and appears on the menu in gray or a lightened shade of the menu-text color.	HELP	Identifies a help item.	INACTIVE	Item name is displayed but it cannot be selected.	MENUBARBREAK	Same as <u>MF_MENUBREAK</u> except that for pop-up menus, it separates the new column from the old column with a vertical line.	MENUBREAK	Places the menu item on a new line for static menu-bar items. For pop-up menus, it places the menu item in a new column with no dividing line between the columns.
Option	Description														
CHECKED	Item has a check mark next to it.														
GRAYED	Item name is initially inactive and appears on the menu in gray or a lightened shade of the menu-text color.														
HELP	Identifies a help item.														
INACTIVE	Item name is displayed but it cannot be selected.														
MENUBARBREAK	Same as <u>MF_MENUBREAK</u> except that for pop-up menus, it separates the new column from the old column with a vertical line.														
MENUBREAK	Places the menu item on a new line for static menu-bar items. For pop-up menus, it places the menu item in a new column with no dividing line between the columns.														

The INACTIVE and GRAYED options cannot be used together.

### Example

The following example demonstrates the usage of the MENUITEM statement:

```
MENUITEM "&Alpha", 1, CHECKED, GRAYED
MENUITEM "&Beta", 2
```

See Also

MENU, POPUP

## POPUP (2.x)

POPUP text, [optionlist]

```
BEGIN
  item-definitions
  :
  :
  :
END
```

The POPUP statement marks the beginning of the definition of a pop-up menu. A pop-up menu (which is also known as a drop-down menu) is a special menu item that displays a sublist of menu items when it is selected.

Parameter	Description												
text	Specifies the name of the pop-up menu. This string must be enclosed in double quotation marks.												
optionlist	Specifies one or more predefined menu options that specify the appearance of the menu item. The menu options follow: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>CHECKED</td><td>Item has a check mark next to it. This option is not valid for a top-level pop-up menu.</td></tr><tr><td>GRAYED</td><td>Item name is initially inactive and appears on the menu in gray or a lightened shade of the menu-text color.</td></tr><tr><td>INACTIVE</td><td>Item name is displayed but it cannot be selected.</td></tr><tr><td>MENUBARBREAK</td><td>Same as <code>MF_MENUBREAK</code> except that for pop-up menus, it separates the new column from the old column with a vertical line.</td></tr><tr><td>MENUBREAK</td><td>Places the menu item on a new line for static menu-bar items. For pop-up menus, it places the menu item in a new column with no dividing line between the columns.</td></tr></tbody></table> The options can be combined using the bitwise OR operator. The INACTIVE and GRAYED options cannot be used together.	Option	Description	CHECKED	Item has a check mark next to it. This option is not valid for a top-level pop-up menu.	GRAYED	Item name is initially inactive and appears on the menu in gray or a lightened shade of the menu-text color.	INACTIVE	Item name is displayed but it cannot be selected.	MENUBARBREAK	Same as <code>MF_MENUBREAK</code> except that for pop-up menus, it separates the new column from the old column with a vertical line.	MENUBREAK	Places the menu item on a new line for static menu-bar items. For pop-up menus, it places the menu item in a new column with no dividing line between the columns.
Option	Description												
CHECKED	Item has a check mark next to it. This option is not valid for a top-level pop-up menu.												
GRAYED	Item name is initially inactive and appears on the menu in gray or a lightened shade of the menu-text color.												
INACTIVE	Item name is displayed but it cannot be selected.												
MENUBARBREAK	Same as <code>MF_MENUBREAK</code> except that for pop-up menus, it separates the new column from the old column with a vertical line.												
MENUBREAK	Places the menu item on a new line for static menu-bar items. For pop-up menus, it places the menu item in a new column with no dividing line between the columns.												

### Example

The following example demonstrates the usage of the POPUP statement:

```
chem MENU
BEGIN
  POPUP "&Elements"
  BEGIN
    MENUITEM "&Oxygen", 200
    MENUITEM "&Carbon", 201, CHECKED
    MENUITEM "&Hydrogen", 202
    MENUITEM "&Sulfur", 203
    MENUITEM "Ch&lorine", 204
  END

  POPUP "&Compounds"
  BEGIN
    POPUP "&Sugars"
    BEGIN
      MENUITEM "&Glucose", 301
      MENUITEM "&Sucrose", 302, CHECKED
      MENUITEM "&Lactose", 303, MENUBREAK
      MENUITEM "&Fructose", 304
    END

    POPUP "&Acids"
    BEGIN
      "&Hydrochloric", 401
```

"&Sulfuric", 402  
END

END

END

See Also

MENU, MENUITEM

## PUSHBUTTON (2.x)

PUSHBUTTON text, id, x, y, width, height, [style]

The PUSHBUTTON statement creates a push-button control. The control is a round-cornered rectangle containing the given text. The control sends a message to its parent whenever the user chooses the control.

Parameter	Description
text	Specifies text that is centered in the rectangular area of the control. This parameter must contain zero or more characters enclosed in double quotation marks. Character values must be in the range 1 through 255. If a double quotation mark is required in the text, you must include the double quotation mark twice. An ampersand (&) character in the text indicates that the following character is used as a mnemonic character for the control. When the control is displayed, the ampersand is not shown but the mnemonic character is underlined. The user can choose the control by pressing the key corresponding to the underlined mnemonic character.
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range.
x	Specifies the x-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box containing the pushbutton.
y	Specifies the y-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box containing the pushbutton.
width	Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range. The width units are 1/4 of the dialog base width unit.
height	Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range. The height units are 1/8 of the dialog base height unit.
style	This optional parameter specifies styles for the pushbutton, which can be a combination of the BS_PUSHBUTTON style and the following styles: <u>WS_TABSTOP</u> , <u>WS_DISABLED</u> , and <u>WS_GROUP</u> .

### Comments

The current dialog base units are computed from the height and width of the current system font. The GetDialogBaseUnits function returns the dialog base units in pixels. The coordinates are relative to the origin of the dialog box.

The default style for PUSHBUTTON is BS\_PUSHBUTTON and WS\_TABSTOP.

### Example

The following example demonstrates the usage of the PUSHBUTTON statement:

```
PUSHBUTTON "ON", 7, 10, 10, 20, 10
```

### See Also

GetDialogBaseUnits

## RADIOBUTTON (2.x)

RADIOBUTTON text, id, x, y, width, height, [style]

The RADIOBUTTON statement creates a radio-button control. The control is a small circle that has the given text displayed next to it, typically to its right. The control highlights the circle and sends a message to its parent window when the user selects the button. The control removes the highlight and sends a message when the button is next selected.

Parameter	Description
text	Specifies text that is centered in the rectangular area of the control. This parameter must contain zero or more characters enclosed in double quotation marks. Character values must be in the range 1 through 255. If a double quotation mark is required in the text, you must include the double quotation mark twice. An ampersand (&) character in the text indicates that the following character is used as a mnemonic character for the control. When the control is displayed, the ampersand is not shown, but the mnemonic character is underlined. The user can choose the control by pressing the key corresponding to the underlined mnemonic character.
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range.
x	Specifies the x-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box containing the radio button.
y	Specifies the y-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box containing the radio button.
width	Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range. The width is in dialog units.
height	Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range. The height is in dialog units.
style	This optional parameter specifies styles for the radio button, which can be a combination of BUTTON-class styles (see Button styles) and the following styles: <u>WS_TABSTOP</u> , <u>WS_DISABLED</u> , and <u>WS_GROUP</u> .

### Comments

Horizontal dialog units are 1/4 of the dialog base width unit. Vertical units are 1/8 of the dialog base height unit. The current dialog base units are computed from the height and width of the current system font. The GetDialogBaseUnits function returns the dialog base units in pixels. The coordinates are relative to the origin of the dialog box.

The default style for RADIOBUTTON is BS\_RADIOBUTTON and WS\_TABSTOP.

The following example demonstrates the usage of the RADIOBUTTON statement:

```
RADIOBUTTON "Italic", 100, 10, 10, 40, 10
```

See Also

GetDialogBaseUnits

## RCDATA (2.x)

nameID RCDATA [load-option] [mem-option]

```
BEGIN
  raw-data
  .
  .
  .
END
```

The RCDATA statement defines a raw data resource for an application. Raw data resources permit the inclusion of binary data directly in the executable file.

Parameter	Description								
nameID	Specifies either a unique name or an integer value that identifies the resource.								
load-option	Specifies when the resource is to be loaded. It takes one of the following keywords: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>PRELOAD</td><td>Resource is loaded immediately.</td></tr><tr><td>LOADONCALL</td><td>Resource is loaded when called. This is the default option.</td></tr></tbody></table>	Option	Description	PRELOAD	Resource is loaded immediately.	LOADONCALL	Resource is loaded when called. This is the default option.		
Option	Description								
PRELOAD	Resource is loaded immediately.								
LOADONCALL	Resource is loaded when called. This is the default option.								
mem-option	Specifies whether the resource is fixed or movable and whether it is discardable. This optional parameter takes one or more of the following keywords: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td><u>FIXED</u></td><td>Resource remains at a fixed memory location.</td></tr><tr><td>MOVEABLE</td><td>Resource can be moved if necessary in order to compact memory.</td></tr><tr><td>DISCARDABLE</td><td>Resource can be discarded if no longer needed.</td></tr></tbody></table> The default memory option is MOVEABLE and DISCARDABLE.	Option	Description	<u>FIXED</u>	Resource remains at a fixed memory location.	MOVEABLE	Resource can be moved if necessary in order to compact memory.	DISCARDABLE	Resource can be discarded if no longer needed.
Option	Description								
<u>FIXED</u>	Resource remains at a fixed memory location.								
MOVEABLE	Resource can be moved if necessary in order to compact memory.								
DISCARDABLE	Resource can be discarded if no longer needed.								
raw-data	Specifies one or more integers and strings. Integers can be in decimal, octal, or hexadecimal format.								

### Example

The following example demonstrates the usage of the RCDATA statement:

```
resname RCDATA
BEGIN
  "Here is a data string\0", /* A string. Note: explicitly
  null-terminated */
  1024, /* int */
  0x029a, /* hex int */
  0o733, /* octal int */
  "\07" /* octal byte */
END
```



## RTEXT (2.x)

RTEXT text, id, x, y, width, height, [style]

The RTEXT statement creates a right-aligned text control. The control is a simple rectangle displaying the given text right-aligned in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next line.

Parameter	Description
text	Specifies text that is aligned on the right side of the rectangular area of the control. This parameter must contain zero or more characters enclosed in double quotation marks. Character values must be in the range 1 through 255. If a double quotation mark is required in the text, you must include the double quotation mark twice. An ampersand (&) character in the text indicates that the following character is used as a mnemonic character for the control. When the control is displayed, the ampersand is not shown but the mnemonic character is underlined. The user can choose the control by pressing the key corresponding to the underlined mnemonic character.
id	Specifies the control identifier. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range.
x	Specifies the x-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box containing the text control.
y	Specifies the y-coordinate of the upper-left corner of the control. This value must be an integer in the range 0 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range. The coordinate is assumed to be in dialog units and is relative to the origin of the dialog box containing the text control.
width	Specifies the width of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range. The width is in dialog units.
height	Specifies the height of the control. This value must be an integer in the range 1 through 65,535 or an expression consisting of integers and the addition (+) operator that evaluates to a value in that range. The height is in dialog units.
style	This optional parameter specifies styles for the text control, which can be any combination of the following: <u>WS_TABSTOP</u> and <u>WS_GROUP</u> .

### Comments

Horizontal dialog units are 1/4 of the dialog base width unit. Vertical units are 1/8 of the dialog base height unit. The current dialog base units are computed from the height and width of the current system font. The GetDialogBaseUnits function returns the dialog base units in pixels. The coordinates are relative to the origin of the dialog box.

The default style for RTEXT is SS\_RIGHT and WS\_GROUP.

### Example

The following example demonstrates the usage of the RTEXT statement:

```
RTEXT "Number of Messages", 4, 30, 50, 100, 10
```

See Also

CONTROL, CTEXT, DIALOG, LTEXT

## SCROLLBAR (2.x)

SCROLLBAR id, x, y, width, height, [style]

The SCROLLBAR statement creates a scroll-bar control. The control is a rectangle that contains a scroll box and has direction arrows at both ends. The scroll-bar control sends a notification message to its parent whenever the user clicks the mouse in the control. The parent is responsible for updating the scroll-box position. Scroll-bar controls can be positioned anywhere in a window and used whenever needed to provide scrolling input.

Parameter	Description
id	Identifies the control. This parameter takes a unique integer value.
x	Specifies the x-coordinate of the upper-left corner of the control in dialog units relative to the origin of the dialog box. The horizontal units are 1/4 of the dialog base width unit.
y	Specifies the y-coordinate of the upper-left corner of the control in dialog units relative to the origin of the dialog box. The vertical units are 1/8 of the dialog base height unit.
width	Specifies the width of the control. The width units are 1/4 of the dialog base width unit.
height	Specifies the height of the control. The height units are 1/8 of the dialog base height unit.
style	Specifies a combination (or none) of the following styles: <u>WS_TABSTOP</u> , <u>WS_GROUP</u> , and <u>WS_DISABLED</u> . In addition to these styles, the style parameter may contain a combination (or none) of the SCROLLBAR-class styles. Styles can be combined by using the bitwise OR operator.

### Comments

The x, y, width, and height parameters can use the addition operator (+) for relative positioning. For example, "15 + 6" can be used for the x parameter.

The default style for SCROLLBAR is SBS\_HORZ.

The current dialog base units are computed from the height and width of the current system font. The GetDialogBaseUnits function returns the dialog base units in pixels.

### Example

The following example demonstrates the usage of the SCROLLBAR statement:

```
SCROLLBAR 999, 25, 30, 10, 100
```

## SEPARATOR (2.x)

### MENUITEM SEPARATOR

The MENUITEM SEPARATOR form of the MENUITEM statement creates an inactive menu item that serves as a dividing bar between two active menu items in a pop-up menu.

#### Example

The following example demonstrates the usage of the MENUITEM SEPARATOR statement:

```
MENUITEM "&Roman", 206  
MENUITEM SEPARATOR  
MENUITEM "&20 Point", 301
```

## STRINGTABLE (2.x)

STRINGTABLE [load-option] [mem-option]

```
BEGIN
    stringID string
    .
    .
    .
END
```

The STRINGTABLE statement defines one or more string resources for an application. String resources are simply null-terminated ASCII strings that can be loaded when needed from the executable file, using the LoadString function.

Parameter	Description								
load-option	Specifies when the resource is to be loaded. This optional parameter must be one of the following keywords: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>PRELOAD</td><td>Resource is loaded immediately.</td></tr><tr><td>LOADONCALL</td><td>Resource is loaded when called. This is the default option.</td></tr></tbody></table>	Option	Description	PRELOAD	Resource is loaded immediately.	LOADONCALL	Resource is loaded when called. This is the default option.		
Option	Description								
PRELOAD	Resource is loaded immediately.								
LOADONCALL	Resource is loaded when called. This is the default option.								
mem-option	Specifies whether the resource is fixed or movable and whether or not it is discardable. This optional parameter can be one of the following keywords: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>FIXED</td><td>Resource remains at a fixed memory location.</td></tr><tr><td>MOVEABLE</td><td>Resource can be moved if necessary in order to compact memory.</td></tr><tr><td>DISCARDABLE</td><td>Resource can be discarded if no longer needed.</td></tr></tbody></table> The default is MOVEABLE and DISCARDABLE.	Option	Description	FIXED	Resource remains at a fixed memory location.	MOVEABLE	Resource can be moved if necessary in order to compact memory.	DISCARDABLE	Resource can be discarded if no longer needed.
Option	Description								
FIXED	Resource remains at a fixed memory location.								
MOVEABLE	Resource can be moved if necessary in order to compact memory.								
DISCARDABLE	Resource can be discarded if no longer needed.								
stringID	Specifies an integer value that identifies the resource.								
string	Specifies one or more ASCII strings, enclosed in double quotation marks. The string must be no longer than 255 characters and must occupy a single line in the source file. To add a carriage return to the string, use this character sequence: \012. For example, "Line one\012Line two" would define a string that would be displayed as follows:  Line one Line two								

### Comments

Grouping strings in separate segments allows all related strings to be read in at one time and discarded together. When possible, an application should make the table movable and discardable. The Resource Compiler allocates 16 strings per segment and uses the identifier value to determine which segment is to contain the string. Strings with the same upper-12 bits in their identifiers are placed in the same segment.

### Example

The following example demonstrates the usage of the STRINGTABLE statement:

```
#define IDS_HELLO    1
#define IDS_GOODBYE 2

STRINGTABLE
BEGIN
    IDS_HELLO,    "Hello"
    IDS_GOODBYE, "Goodbye"
END
```

## STYLE (2.x)

### STYLE style

The STYLE statement defines the window style of the dialog box. The window style specifies whether the box is a pop-up or a child window. The default style has the following attributes: WS\_POPUP, WS\_BORDER, and WS\_SYSMENU.

Parameter	Description																												
style	Specifies the window style. This parameter takes an integer value or predefined name. The following lists the predefined styles:																												
	<table><thead><tr><th>Style</th><th>Meaning</th></tr></thead><tbody><tr><td><u>DS_LOCALEDIT</u></td><td>Specifies that edit controls in the dialog box will use memory in the application's data segment. By default, all edit controls in dialog boxes use memory outside the application's data segment. This feature can be suppressed by adding the <u>DS_LOCALEDIT</u> flag to the STYLE command for the dialog box. If this flag is not used, <u>EM_GETHANDLE</u> and <u>EM_SETHANDLE</u> messages must not be used since the storage for the control is not in the application's data segment. This feature does not affect edit controls created outside of dialog boxes.</td></tr><tr><td><u>DS_MODALFRAME</u></td><td>Creates a dialog box with a modal dialog box frame that can be combined with a title bar and System menu by specifying the <u>WS_CAPTION</u> and <u>WS_SYSMENU</u> styles.</td></tr><tr><td><u>DS_NOIDLEMSG</u></td><td>Suppresses <u>WM_ENTERIDLE</u> messages that Windows would otherwise send to the owner of the dialog box while the dialog box is displayed.</td></tr><tr><td><u>DS_SYSMODAL</u></td><td>Creates a system-modal dialog box.</td></tr><tr><td><u>WS_BORDER</u></td><td>Creates a window that has a border.</td></tr><tr><td><u>WS_CAPTION</u></td><td>Creates a window that has a title bar (implies the <u>WS_BORDER</u> style).</td></tr><tr><td><u>WS_CHILD</u></td><td>Creates a child window. It cannot be used with the <u>WS_POPUP</u> style.</td></tr><tr><td><u>WS_CHILDWINDOW</u></td><td>Creates a child window that has the <u>WS_CHILD</u> style.</td></tr><tr><td><u>WS_CLIPCHILDREN</u></td><td>Excludes the area occupied by child windows when drawing within the parent window. Used when creating the parent window.</td></tr><tr><td><u>WS_CLIPSIBLINGS</u></td><td>Clips child windows relative to each other; that is, when a particular child window receives a <u>WM_PAINT</u> message, this style clips all other top-level child windows out of the region of the child window to be updated. (If the <u>WS_CLIPSIBLINGS</u> style is not given and child windows overlap, it is possible, when drawing in the client area of a child window, to draw in the client area of a neighboring child window.) For use with the <u>WS_CHILD</u> style only.</td></tr><tr><td><u>WS_DISABLED</u></td><td>Creates a window that is initially disabled.</td></tr><tr><td><u>WS_DLGMFRAME</u></td><td>Creates a window with a modal dialog box frame but no title.</td></tr><tr><td><u>WS_GROUP</u></td><td>Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls defined with the <u>WS_GROUP</u> style after the first control belong to the same group. The next control with the <u>WS_GROUP</u> style ends the style</td></tr></tbody></table>	Style	Meaning	<u>DS_LOCALEDIT</u>	Specifies that edit controls in the dialog box will use memory in the application's data segment. By default, all edit controls in dialog boxes use memory outside the application's data segment. This feature can be suppressed by adding the <u>DS_LOCALEDIT</u> flag to the STYLE command for the dialog box. If this flag is not used, <u>EM_GETHANDLE</u> and <u>EM_SETHANDLE</u> messages must not be used since the storage for the control is not in the application's data segment. This feature does not affect edit controls created outside of dialog boxes.	<u>DS_MODALFRAME</u>	Creates a dialog box with a modal dialog box frame that can be combined with a title bar and System menu by specifying the <u>WS_CAPTION</u> and <u>WS_SYSMENU</u> styles.	<u>DS_NOIDLEMSG</u>	Suppresses <u>WM_ENTERIDLE</u> messages that Windows would otherwise send to the owner of the dialog box while the dialog box is displayed.	<u>DS_SYSMODAL</u>	Creates a system-modal dialog box.	<u>WS_BORDER</u>	Creates a window that has a border.	<u>WS_CAPTION</u>	Creates a window that has a title bar (implies the <u>WS_BORDER</u> style).	<u>WS_CHILD</u>	Creates a child window. It cannot be used with the <u>WS_POPUP</u> style.	<u>WS_CHILDWINDOW</u>	Creates a child window that has the <u>WS_CHILD</u> style.	<u>WS_CLIPCHILDREN</u>	Excludes the area occupied by child windows when drawing within the parent window. Used when creating the parent window.	<u>WS_CLIPSIBLINGS</u>	Clips child windows relative to each other; that is, when a particular child window receives a <u>WM_PAINT</u> message, this style clips all other top-level child windows out of the region of the child window to be updated. (If the <u>WS_CLIPSIBLINGS</u> style is not given and child windows overlap, it is possible, when drawing in the client area of a child window, to draw in the client area of a neighboring child window.) For use with the <u>WS_CHILD</u> style only.	<u>WS_DISABLED</u>	Creates a window that is initially disabled.	<u>WS_DLGMFRAME</u>	Creates a window with a modal dialog box frame but no title.	<u>WS_GROUP</u>	Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls defined with the <u>WS_GROUP</u> style after the first control belong to the same group. The next control with the <u>WS_GROUP</u> style ends the style
Style	Meaning																												
<u>DS_LOCALEDIT</u>	Specifies that edit controls in the dialog box will use memory in the application's data segment. By default, all edit controls in dialog boxes use memory outside the application's data segment. This feature can be suppressed by adding the <u>DS_LOCALEDIT</u> flag to the STYLE command for the dialog box. If this flag is not used, <u>EM_GETHANDLE</u> and <u>EM_SETHANDLE</u> messages must not be used since the storage for the control is not in the application's data segment. This feature does not affect edit controls created outside of dialog boxes.																												
<u>DS_MODALFRAME</u>	Creates a dialog box with a modal dialog box frame that can be combined with a title bar and System menu by specifying the <u>WS_CAPTION</u> and <u>WS_SYSMENU</u> styles.																												
<u>DS_NOIDLEMSG</u>	Suppresses <u>WM_ENTERIDLE</u> messages that Windows would otherwise send to the owner of the dialog box while the dialog box is displayed.																												
<u>DS_SYSMODAL</u>	Creates a system-modal dialog box.																												
<u>WS_BORDER</u>	Creates a window that has a border.																												
<u>WS_CAPTION</u>	Creates a window that has a title bar (implies the <u>WS_BORDER</u> style).																												
<u>WS_CHILD</u>	Creates a child window. It cannot be used with the <u>WS_POPUP</u> style.																												
<u>WS_CHILDWINDOW</u>	Creates a child window that has the <u>WS_CHILD</u> style.																												
<u>WS_CLIPCHILDREN</u>	Excludes the area occupied by child windows when drawing within the parent window. Used when creating the parent window.																												
<u>WS_CLIPSIBLINGS</u>	Clips child windows relative to each other; that is, when a particular child window receives a <u>WM_PAINT</u> message, this style clips all other top-level child windows out of the region of the child window to be updated. (If the <u>WS_CLIPSIBLINGS</u> style is not given and child windows overlap, it is possible, when drawing in the client area of a child window, to draw in the client area of a neighboring child window.) For use with the <u>WS_CHILD</u> style only.																												
<u>WS_DISABLED</u>	Creates a window that is initially disabled.																												
<u>WS_DLGMFRAME</u>	Creates a window with a modal dialog box frame but no title.																												
<u>WS_GROUP</u>	Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls defined with the <u>WS_GROUP</u> style after the first control belong to the same group. The next control with the <u>WS_GROUP</u> style ends the style																												

	group and starts the next group (that is, one group ends where the next begins). This style is valid only for controls.
WS_HSCROLL	Creates a window that has a horizontal scroll bar.
WS_ICONIC	Creates a window that is initially iconic. For use with the <u>WS_OVERLAPPED</u> style only.
WS_MAXIMIZE	Creates a window of maximum size.
WS_MAXIMIZEBOX	Creates a window that has a Maximize box.
WS_MINIMIZE	Creates a window of minimum size.
WS_MINIMIZEBOX	Creates a window that has a Minimize box.
WS_OVERLAPPED	Creates an overlapped window. An overlapped window has a caption and a border.
WS_OVERLAPPEDWINDOW	Creates an overlapped window having the <u>WS_OVERLAPPED</u> , <u>WS_CAPTION</u> , <u>WS_SYSMENU</u> , <u>WS_THICKFRAME</u> , <u>WS_MINIMIZEBOX</u> , and <u>WS_MAXIMIZEBOX</u> styles.
WS_POPUP	Creates a pop-up window. It cannot be used with the <u>WS_CHILD</u> style.
WS_POPUPWINDOW	Creates a pop-up window that has the <u>WS_POPUP</u> , <u>WS_BORDER</u> , and <u>WS_SYSMENU</u> styles. The <u>WS_CAPTION</u> style must be combined with the <u>WS_POPUPWINDOW</u> style to make the System menu visible.
WS_SIZEBOX	Creates a window that has a size box. Used only for windows with a title bar or with vertical and horizontal scroll bars.
WS_SYSMENU	Creates a window that has a System-menu box in its title bar. Used only for windows with title bars. If used with a child window, this style creates a Close box instead of a System-menu box.
WS_TABSTOP	Specifies one of any number of controls through which the user can move by using the TAB key. The TAB key moves the user to the next control specified by the <u>WS_TABSTOP</u> style. This style is valid only for controls.
WS_THICKFRAME	Creates a window with a thick frame that can be used to size the window.
WS_VISIBLE	Creates a window that is initially visible. This applies to overlapping and pop-up windows. For overlapping windows, the y parameter is used as a parameter for the <u>ShowWindow</u> function.
WS_VSCROLL	Creates a window that has a vertical scroll bar.

#### Comments

If the predefined names are used, the #include directive must be used so that the WINDOWS.H file will be included in the resource script.

## DS\_LOCALEEDIT

Specifies that edit controls in the dialog box will use memory in the application's data segment. By default, all edit controls in dialog boxes use memory outside the application's data segment. This feature can be suppressed by adding the DS\_LOCALEEDIT flag to the STYLE command for the dialog box. If this flag is not used, EM\_GETHANDLE and EM\_SETHANDLE messages must not be used since the storage for the control is not in the application's data segment. This feature does not affect edit controls created outside of dialog boxes.

## DS\_MODALFRAME

Creates a dialog box with a modal dialog box frame that can be combined with a title bar and System menu by specifying the WS\_CAPTION and WS\_SYSMENU styles.



DS\_NOIDLEMSG

Suppresses WM\_ENTERIDLE messages that Windows would otherwise send to the owner of the dialog box while the dialog box is displayed.

DS\_SYSMODAL

Creates a system-modal dialog box.

`WS_BORDER`

Creates a window that has a border.

**WS\_CAPTION**

Creates a window that has a title bar (implies the WS\_BORDER style).

WS\_CHILD

Creates a child window. It cannot be used with the WS\_POPUP style.

WS\_CHILDWINDOW

Creates a child window that has the WS\_CHILD style.

WS\_CLIPCHILDREN

Excludes the area occupied by child windows when drawing within the parent window. Used when creating the parent window.

## WS\_CLIPSIBLINGS

Clips child windows relative to each other; that is, when a particular child window receives a WM\_PAINT message, this style clips all other top-level child windows out of the region of the child window to be updated. (If the WS\_CLIPSIBLINGS style is not given and child windows overlap, it is possible, when drawing in the client area of a child window, to draw in the client area of a neighboring child window.) For use with the WS\_CHILD style only.



**WS\_DISABLED**

Creates a window that is initially disabled.

`WS_DLGFRAME`

Creates a window with a modal dialog box frame but no title.

## WS\_GROUP

Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls defined with the WS\_GROUP style after the first control belong to the same group. The next control with the WS\_GROUP style ends the style group and starts the next group (that is, one group ends where the next begins). This style is valid only for controls.

`WS_HSCROLL`

Creates a window that has a horizontal scroll bar.

WS\_ICONIC

Creates a window that is initially iconic. For use with the WS\_OVERLAPPED style only.

WS\_MAXIMIZE

Creates a window of maximum size.

WS\_MAXIMIZEBOX

Creates a window that has a Maximize box.

WS\_MINIMIZE

Creates a window of minimum size.



**WS\_MINIMIZEBOX**

Creates a window that has a Minimize box.

`WS_OVERLAPPED`

Creates an overlapped window. An overlapped window has a caption and a border.

WS\_OVERLAPPEDWINDOW

Creates an overlapped window having the WS\_OVERLAPPED, WS\_CAPTION, WS\_SYSMENU, WS\_THICKFRAME, WS\_MINIMIZEBOX, and WS\_MAXIMIZEBOX styles.

WS\_POPUP

Creates a pop-up window. It cannot be used with the WS\_CHILD style.

## WS\_POPUPWINDOW

Creates a pop-up window that has the WS\_POPUP, WS\_BORDER, and WS\_SYSMENU styles. The WS\_CAPTION style must be combined with the WS\_POPUPWINDOW style to make the System menu visible.

## WS\_SIZEBOX

Creates a window that has a size box. Used only for windows with a title bar or with vertical and horizontal scroll bars.

## WS\_SYSMENU

Creates a window that has a System-menu box in its title bar. Used only for windows with title bars. If used with a child window, this style creates a Close box instead of a System-menu box.

## WS\_TABSTOP

Specifies one of any number of controls through which the user can move by using the TAB key. The TAB key moves the user to the next control specified by the WS\_TABSTOP style. This style is valid only for controls.



**WS\_THICKFRAME**

Creates a window with a thick frame that can be used to size the window.

## WS\_VISIBLE

Creates a window that is initially visible. This applies to overlapping and pop-up windows. For overlapping windows, the y parameter is used as a parameter for the ShowWindow function.

`WS_VSCROLL`

Creates a window that has a vertical scroll bar.

## #undef (2.x)

#undef name

The #undef directive removes the current definition of the specified name. All subsequent occurrences of the name are processed without replacement.

Parameter	Description
name	Specifies the name to be removed. This value is any combination of letters, digits, and punctuation.

### Example

This example removes the definitions for the names "nonzero" and "USERCLASS":

```
#undefnonzero  
#undefUSERCLASS
```

See Also

#define

## User-Defined (3.0)

nameID typeID [load-option] [mem-option] filename

nameID typeID [load-option] [mem-option]

```
BEGIN
  raw-data
  .
  .
  .
END
```

A user-defined resource statement specifies a resource that contains application-specific data. The data can have any format and can be defined either as the content of a given file (if the filename parameter is given) or as a series of numbers or strings (if the raw-data parameter is given).

Parameter	Description								
nameID	Specifies either a unique name or an integer that identifies the resource.								
typeID	Specifies either a unique name or an integer that identifies the resource type. If a number is given, it must be greater than 255. The numbers 1 through 255 are reserved for existing and future predefined resource types.								
load-option	Specifies when the resource is to be loaded. The parameter must be one of the following options: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td>PRELOAD</td><td>Resource is loaded immediately.</td></tr><tr><td>LOADONCALL</td><td>Resource is loaded when called. This is the default option.</td></tr></tbody></table>	Option	Description	PRELOAD	Resource is loaded immediately.	LOADONCALL	Resource is loaded when called. This is the default option.		
Option	Description								
PRELOAD	Resource is loaded immediately.								
LOADONCALL	Resource is loaded when called. This is the default option.								
mem-option	Specifies whether the resource is fixed or movable and whether it is discardable. The parameter must be one of the following options: <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td><u>FIXED</u></td><td>Resource remains at a fixed memory location.</td></tr><tr><td>MOVEABLE</td><td>Resource can be moved if necessary in order to compact memory.</td></tr><tr><td>DISCARDABLE</td><td>Resource can be discarded if no longer needed.</td></tr></tbody></table> The default is MOVEABLE and DISCARDABLE for cursor, icon, and font resources. The default for bitmap resources is MOVEABLE.	Option	Description	<u>FIXED</u>	Resource remains at a fixed memory location.	MOVEABLE	Resource can be moved if necessary in order to compact memory.	DISCARDABLE	Resource can be discarded if no longer needed.
Option	Description								
<u>FIXED</u>	Resource remains at a fixed memory location.								
MOVEABLE	Resource can be moved if necessary in order to compact memory.								
DISCARDABLE	Resource can be discarded if no longer needed.								
filename	Specifies the name of the file that contains the resource data. The parameter must be a valid MS-DOS filename; it must be a full path if the file is not in the current working directory.								
raw-data	Specifies one or more integers and strings. Integers can be in decimal, octal, or hexadecimal format.								

### Example

The following example shows several user-defined statements:

```
array MYRES data.res
14 300custom.res
18 MYRES2
BEGIN
  "Here is a data string\0", /* A string. Note: explicitly
  null-terminated */
  1024, /* int */
  0x029a, /* hex int */
  0o733, /* octal int */
  "\07" /* octal byte */
END
```

## VERSIONINFO (3.1)

versionID VERSIONINFO fixed-info

```
BEGIN
    block-statement
    .
    .
    .
END
```

The VERSIONINFO statement creates a version-information resource. The resource contains such information about the file as its version number, its intended operating system, and its original filename. The resource is intended to be used with the File Installation library functions.

Parameter	Description																
versionID	Specifies the version-information resource identifier. This value must be 1.																
fixed-info	Specifies the version information, such as the file version and the intended operating system. This parameter consists of the following statements: <table><thead><tr><th>Statement</th><th>Description</th></tr></thead><tbody><tr><td>FILEVERSION version</td><td>Specifies the binary version number for the file. The version consists of two 32-bit integers, defined by four 16-bit integers. For example, "FILEVERSION 3,10,0,61" is translated into two doublewords: 0x0003000a and 0x0000003d, in that order. Therefore, if version is defined by the doublewords dw1 and dw2, they need to appear in the FILEVERSION statement as follows: HIWORD(dw1), LOWORD(dw1), HIWORD(dw2), LOWORD(dw2).</td></tr><tr><td>PRODUCTVERSION version</td><td>Specifies the binary version number for the product with which the file is distributed. The version parameter is two 32-bit integers, defined by four 16-bit integers. For more information about version, see the FILEVERSION description.</td></tr><tr><td>FILEFLAGSMASK fileflagsmask</td><td>Specifies which bits in the FILEFLAGS statement are valid. If a bit is set, the corresponding bit in FILEFLAGS is valid.</td></tr><tr><td>FILEFLAGS fileflags</td><td>Specifies the Boolean attributes of the file. The fileflags parameter must be the combination of all the file flags that are valid at compile time. For Windows 3.1, this value is 0x3f.</td></tr><tr><td>FILEOS fileos</td><td>Specifies the operating system for which this file was designed. The fileos parameter can be one of the operating system values given in the Comments section.</td></tr><tr><td>FILETYPE filetype</td><td>Specifies the general type of file. The filetype parameter can be one of the file type values listed in the Comments section.</td></tr><tr><td>FILESUBTYPE subtype</td><td>Specifies the function of the file. The subtype parameter is zero unless the type parameter in the FILETYPE statement is VFT_DRV, VFT_FONT, or VFT_VXD. For a list of file</td></tr></tbody></table>	Statement	Description	FILEVERSION version	Specifies the binary version number for the file. The version consists of two 32-bit integers, defined by four 16-bit integers. For example, "FILEVERSION 3,10,0,61" is translated into two doublewords: 0x0003000a and 0x0000003d, in that order. Therefore, if version is defined by the doublewords dw1 and dw2, they need to appear in the FILEVERSION statement as follows: HIWORD(dw1), LOWORD(dw1), HIWORD(dw2), LOWORD(dw2).	PRODUCTVERSION version	Specifies the binary version number for the product with which the file is distributed. The version parameter is two 32-bit integers, defined by four 16-bit integers. For more information about version, see the FILEVERSION description.	FILEFLAGSMASK fileflagsmask	Specifies which bits in the FILEFLAGS statement are valid. If a bit is set, the corresponding bit in FILEFLAGS is valid.	FILEFLAGS fileflags	Specifies the Boolean attributes of the file. The fileflags parameter must be the combination of all the file flags that are valid at compile time. For Windows 3.1, this value is 0x3f.	FILEOS fileos	Specifies the operating system for which this file was designed. The fileos parameter can be one of the operating system values given in the Comments section.	FILETYPE filetype	Specifies the general type of file. The filetype parameter can be one of the file type values listed in the Comments section.	FILESUBTYPE subtype	Specifies the function of the file. The subtype parameter is zero unless the type parameter in the FILETYPE statement is VFT_DRV, VFT_FONT, or VFT_VXD. For a list of file
Statement	Description																
FILEVERSION version	Specifies the binary version number for the file. The version consists of two 32-bit integers, defined by four 16-bit integers. For example, "FILEVERSION 3,10,0,61" is translated into two doublewords: 0x0003000a and 0x0000003d, in that order. Therefore, if version is defined by the doublewords dw1 and dw2, they need to appear in the FILEVERSION statement as follows: HIWORD(dw1), LOWORD(dw1), HIWORD(dw2), LOWORD(dw2).																
PRODUCTVERSION version	Specifies the binary version number for the product with which the file is distributed. The version parameter is two 32-bit integers, defined by four 16-bit integers. For more information about version, see the FILEVERSION description.																
FILEFLAGSMASK fileflagsmask	Specifies which bits in the FILEFLAGS statement are valid. If a bit is set, the corresponding bit in FILEFLAGS is valid.																
FILEFLAGS fileflags	Specifies the Boolean attributes of the file. The fileflags parameter must be the combination of all the file flags that are valid at compile time. For Windows 3.1, this value is 0x3f.																
FILEOS fileos	Specifies the operating system for which this file was designed. The fileos parameter can be one of the operating system values given in the Comments section.																
FILETYPE filetype	Specifies the general type of file. The filetype parameter can be one of the file type values listed in the Comments section.																
FILESUBTYPE subtype	Specifies the function of the file. The subtype parameter is zero unless the type parameter in the FILETYPE statement is VFT_DRV, VFT_FONT, or VFT_VXD. For a list of file																

subtype values, see the Comments section.

block-statement Specifies one or more version-information blocks. A block can contain string information or variable information.

Comments

To use the constants specified with the VERSIONINFO statement, the VER.H file must be included in the resource-definition file.

The following list describes the parameters used in the VERSIONINFO statement:

Parameter	Description	
fileflags	Specifies a combination of the following values:	
	Value	Meaning
	VS_FF_DEBUG	File contains debugging information or is compiled with debugging features enabled.
	VS_FF_INFOINFERRED	File contains a dynamically created version-information resource. Some of the blocks for the resource may be empty or incorrect. This value is not intended to be used in version-information resources created by using the VERSIONINFO statement.
	VS_FF_PATCHED	File has been modified and is not identical to the original shipping file of the same version number.
	VS_FF_PRERELEASE	File is a development version, not a commercially released product.
	VS_FF_PRIVATEBUILD	File was not built using standard release procedures. If this value is given, the StringFileInfo block must contain a PrivateBuild string.
	VS_FF_SPECIALBUILD	File was built by the original company using standard release procedures but is a variation of the standard file of the same version number. If this value is given, the StringFileInfo block must contain a SpecialBuild string.
fileos	Specifies one of the following values:	
	Value	Meaning
	VOS_UNKNOWN	Operating system for which the file was designed is unknown to Windows.
	VOS_DOS	File was designed for MS-DOS.
	VOS_NT	File was designed for Windows NT.
	VOS_WINDOWS16	File was designed for Windows version 3.0 or later.
	VOS_WINDOWS32	File was designed for 32-bit Windows.
	VOS_DOS_WINDOWS16	File was designed for Windows version 3.0 or later running with MS-DOS.
	VOS_DOS_WINDOWS32	File was designed for 32-bit Windows running with MS-DOS.
	VOS_NT_WINDOWS32	File was designed for 32-bit Windows running with Windows NT.
	The values 0x00002L, 0x00003L, 0x20000L and 0x30000L are reserved.	
filetype	Specifies one of the following values:	
	Value	Meaning
	VFT_UNKNOWN	File type is unknown to Windows.
	VFT_APP	File contains an application.
	VFT_DLL	File contains a dynamic-link library (DLL).
	VFT_DRV	File contains a device driver. If the dwFileType member is VFT_DRV, the dwFileSubtype member contains a more specific description of the driver.
	VFT_FONT	File contains a font. If the dwFileType member is VFT_FONT, the dwFileSubtype member contains a more specific description of the font.

**subtype**  
 VFT\_VXD File contains a virtual device.  
 VFT\_STATIC\_LIB File contains a static-link library.  
 All other values are reserved for use by Microsoft.  
 Specifies additional information about the file type.  
 If the FILETYPE statement specifies VFT\_DRV, this parameter can be one of the following values:

Value	Meaning
VFT2_UNKNOWN	Driver type is unknown to Windows.
VFT2_DRV_COMM	File contains a communications driver.
VFT2_DRV_PRINTER	File contains a printer driver.
VFT2_DRV_KEYBOARD	File contains a keyboard driver.
VFT2_DRV_LANGUAGE	File contains a language driver.
VFT2_DRV_DISPLAY	File contains a display driver.
VFT2_DRV_MOUSE	File contains a mouse driver.
VFT2_DRV_NETWORK	File contains a network driver.
VFT2_DRV_SYSTEM	File contains a system driver.
VFT2_DRV_INSTALLABLE	File contains an installable driver.
VFT2_DRV_SOUND	File contains a sound driver.

If the FILETYPE statement specifies VFT\_FONT, this parameter can be one of the following values:

Value	Meaning
VFT2_UNKNOWN	Font type is unknown to Windows.
VFT2_FONT_RASTER	File contains a raster font.
VFT2_FONT_VECTOR	File contains a vector font.
VFT2_FONT_TRUETYPE	File contains a TrueType font.

If the FILETYPE statement specifies VFT\_VXD, this parameter must be the virtual-device identifier included in the virtual-device control block.  
 All subtype values not listed here are reserved for use by Microsoft.

**langID**  
 Specifies one of the following language identifiers:

Value	Language
0x0401	Arabic
0x0402	Bulgarian
0x0403	Catalan
0x0404	Traditional Chinese
0x0405	Czech
0x0406	Danish
0x0407	German
0x0408	Greek
0x0409	U.S. English
0x040A	Castilian Spanish
0x040B	Finnish
0x040C	French
0x040D	Hebrew
0x040E	Hungarian
0x040F	Icelandic
0x0410	Italian
0x0411	Japanese
0x0412	Korean
0x0413	Dutch
0x0414	Norwegian - Bokmål
0x0415	Polish
0x0416	Brazilian Portuguese
0x0417	Rhaeto-Romanic



0x0418	Romanian
0x0419	Russian
0x041A	Croato-Serbian (Latin)
0x041B	Slovak
0x041C	Albanian
0x041D	Swedish
0x041E	Thai
0x041F	Turkish
0x0420	Urdu
0x0421	Bahasa
0x0804	Simplified Chinese
0x0807	Swiss German
0x0809	U.K. English
0x080A	Mexican Spanish
0x080C	Belgian French
0x0810	Swiss Italian
0x0813	Belgian Dutch
0x0814	Norwegian - Nynorsk
0x0816	Portuguese
0x081A	Serbo-Croatian (Cyrillic)
0x0C0C	Canadian French
0x100C	Swiss French

charsetID Specifies one of the following character-set identifiers:

Value	Character set
0	7-bit ASCII
932	Windows, Japan (Shift - JIS X-0208)
949	Windows, Korea (Shift - KSC 5601)
950	Windows, Taiwan (GB5)
1200	Unicode
1250	Windows, Latin-2 (Eastern European)
1251	Windows, Cyrillic
1252	Windows, Multilingual
1253	Windows, Greek
1254	Windows, Turkish
1255	Windows, Hebrew
1256	Windows, Arabic

string-name Specifies one of the following predefined names:

Name	Value
Comments	Specifies additional information that should be displayed for diagnostic purposes.
CompanyName	Specifies the company that produced the file--for example, "Microsoft Corporation" or "Standard Microsystems Corporation, Inc.". This string is required.
FileDescription	Specifies a file description to be presented to users. This string may be displayed in a list box when the user is choosing files to install--for example, "Keyboard Driver for AT-Style Keyboards" or "Microsoft Word for Windows". This string is required.
FileVersion	Specifies the version number of the file--for example, "3.10" or "5.00.RC2". This string is required.
InternalName	Specifies the internal name of the file, if one exists--for example, a module name if the file is a dynamic-link library. If the file has no internal name, this string should be the original filename, without extension. This string is required.

LegalCopyright	Specifies all copyright notices that apply to the file. This should include the full text of all notices, legal symbols, copyright dates, and so on--for example, "Copyright Microsoft Corporation 1990-1991". This string is optional.
LegalTrademarks	Specifies all trademarks and registered trademarks that apply to the file. This should include the full text of all notices, legal symbols, trademark numbers, and so on--for example, "Windows(TM) is a trademark of Microsoft Corporation". This string is optional.
OriginalFilename	Specifies the original name of the file, not including a path. This information enables an application to determine whether a file has been renamed by a user. The format of the name depends on the file system for which the file was created. This string is required.
PrivateBuild	Specifies information about a private version of the file--for example, "Built by TESTER1 on \TESTBED". This string should be present only if the VS_FF_PRIVATEBUILD flag is set in the dwFileFlags member of the <u>VS_FIXEDFILEINFO</u> structure of the root block.
ProductName	Specifies the name of the product with which the file is distributed--for example, "Microsoft Windows". This string is required.
ProductVersion	Specifies the version of the product with which the file is distributed--for example, "3.10" or "5.00.RC2". This string is required.
SpecialBuild	Specifies how this version of the file differs from the standard version--for example, "Private build for TESTER1 solving mouse problems on M250 and M250E computers". This string should be present only if the VS_FF_SPECIALBUILD flag is set in the dwFileFlags member of the <u>VS_FIXEDFILEINFO</u> structure in the root block.

A string information block has the following form:

```

BLOCK "StringFileInfo"
BEGIN
  BLOCK "lang-charset"
  BEGIN
    VALUE "string-name", "value"
  .
  .
  .
  END
END

```

Following are the parameters in the StringFileInfo block:

lang-charset	Specifies a language and character-set identifier pair. It is a hexadecimal string consisting of the concatenation of the language and character-set identifiers listed earlier in this section.
string-name	Specifies the name of a value in the block and can be one of the predefined names listed earlier in this section.
value	Specifies, as a character string, the value of the corresponding string name. More than one VALUE statement can be given.

A variable information block has the following form:

```

BLOCK "VarFileInfo"
BEGIN
  VALUE "Translation",
  langID, charsetID
.
.
.

```

END

Following are the parameters in the variable information block:

langID            Specifies one of the language identifiers listed earlier in this section.

charsetID        Specifies one of the character-set identifiers listed earlier in this section.

More than one identifier pair can be given, but each pair must be separated from the preceding pair with a comma.

## Resource Statements

<u>ACCELERATORS</u>	Defines accelerator keystroke
<u>BITMAP</u>	Defines a bitmap resource
<u>CAPTION</u>	Defines the title for a dialog box
<u>CHECKBOX</u>	Creates a predefined check-box control
<u>CLASS</u>	Defines the class of a dialog box
<u>COMBOBOX</u>	Creates a combination-box control
<u>CONTROL</u>	Creates a control for a dialog box
<u>CTEXT</u>	Creates a centered text control
<u>CURSOR</u>	Specifies a cursor resource
<u>#define</u>	Assigns a value to a name
<u>DEFPUSHBUTTON</u>	Creates a default push-button control
<u>DIALOG</u>	Defines a dialog window
<u>EDITTEXT</u>	Defines an EDIT control
<u>#elif</u>	Compiles conditionally (else if)
<u>#else</u>	Compiles if conditional directive is false
<u>#endif</u>	Marks the end of an #ifdef block
<u>FONT</u>	Specifies the font in a dialog box
<u>FONT</u>	Specifies a font resource
<u>GROUPBOX</u>	Creates a group box
<u>ICON</u>	Creates an icon control
<u>ICON</u>	Specifies an icon resource
<u>#if</u>	Conditionally compiles if an expression is true
<u>#ifdef</u>	Conditionally compiles if a name is defined
<u>#ifndef</u>	Conditionally compiles if a name is not defined
<u>#include</u>	Includes a header file
<u>LISTBOX</u>	Creates a list-box control
<u>LTEXT</u>	Creates a left-aligned text control
<u>MENU</u>	Defines a dialog box's menu
<u>MENU</u>	Creates a menu
<u>MENUITEM</u>	Defines a menu item
<u>POPUP</u>	Creates a pop-up menu
<u>PUSHBUTTON</u>	Creates a pushbutton control
<u>RADIOBUTTON</u>	Creates a radio-button control
<u>RCDATA</u>	Defines a raw-data resource
<u>RTEXT</u>	Creates a right-aligned text control
<u>SCROLLBAR</u>	Creates a scrollbar control
<u>SEPARATOR</u>	Creates inactive dividing bar in menu
<u>STRINGTABLE</u>	Defines string resources
<u>STYLE</u>	Defines window style of dialog box
<u>#undef</u>	Removes a name definition
<u>User-Defined</u>	User-Defined Resources
<u>VERSIONINFO</u>	Creates a version information resource

## RegCloseKey (3.1)

```
#include shellapi.h  
  
LONG RegCloseKey(hkey)  
HKEY hkey; /* handle of key to close */
```

The `RegCloseKey` function closes a key. Closing a key releases the key's handle. When all keys are closed, the registration database is updated.

Parameter	Description
<code>hkey</code>	Identifies the open key to close.

### Returns

The return value is `ERROR_SUCCESS` if the function is successful. Otherwise, it is an error value.

### Comments

The `RegCloseKey` function should be called only if a key has been opened by either the `RegOpenKey` function or the `RegCreateKey` function. The handle for a given key should not be used after it has been closed, because it may no longer be valid. Key handles should not be left open any longer than necessary.

### Example

The following example uses the `RegCreateKey` function to create the handle of a protocol, uses the `RegSetValue` function to set up the subkeys of the protocol, and then calls `RegCloseKey` to save the information in the database:

```
HKEY hkProtocol;  
  
if (RegCreateKey(HKEY_CLASSES_ROOT, /* root */  
    "NewAppDocument\\protocol\\StdFileEditing", /* protocol string */  
    &hkProtocol) != ERROR_SUCCESS)/* protocol key handle */  
    return FALSE;  
  
RegSetValue(hkProtocol,/* handle of protocol key */  
    "server",/* name of subkey */  
    REG_SZ, /* required */  
    "newapp.exe", /* command to activate server*/  
    10);/* text string size*/  
  
RegSetValue(hkProtocol,/* handle of protocol key */  
    "verb\\0", /* name of subkey */  
    REG_SZ, /* required */  
    "Edit", /* server should edit object */  
    4); /* text string size*/  
  
RegCloseKey(hkProtocol);/* closes protocol key and subkeys */
```

### See Also

[RegCreateKey](#), [RegDeleteKey](#), [RegOpenKey](#), [RegSetValue](#)

## RegCreateKey (3.1)

#include shellapi.h

```
LONG RegCreateKey(hkey, lpszSubKey, lphkResult)
HKEY hkey; /* handle of an open key */
LPCSTR lpszSubKey; /* address of string for subkey to open */
HKEY FAR* lphkResult; /* address of handle of open key */
```

The RegCreateKey function creates the specified key. If the key already exists in the registration database, RegCreateKey opens it.

Parameter	Description
hkey	Identifies an open key (which can be HKEY_CLASSES_ROOT). The key opened or created by the RegCreateKey function is a subkey of the key identified by the hkey parameter. This value should not be NULL.
lpszSubKey	Points to a null-terminated string specifying the subkey to open or create.
lphkResult	Points to the handle of the key that is opened or created.

### Returns

The return value is ERROR\_SUCCESS if the function is successful. Otherwise, it is an error value.

### Comments

An application can create keys that are subordinate to the top level of the database by specifying HKEY\_CLASSES\_ROOT for the hKey parameter. An application can use the RegCreateKey function to create several keys at once. For example, an application could create a subkey four levels deep and the three preceding subkeys by specifying a string of the following form for the lpszSubKey parameter:

```
subkey1\subkey2\subkey3\subkey4
```

### Example

The following example uses the RegCreateKey function to create the handle of a protocol, uses the RegSetValue function to set up the subkeys of the protocol, and then calls RegCloseKey to save the information in the database:

```
HKEY hkProtocol;

if (RegCreateKey(HKEY_CLASSES_ROOT, /* root */
    "NewAppDocument\protocol\StdFileEditing", /* protocol string */
    &hkProtocol) != ERROR_SUCCESS)/* protocol key handle */
    return FALSE;

RegSetValue(hkProtocol,/* handle of protocol key */
    "server",/* name of subkey */
    REG_SZ, /* required */
    "newapp.exe", /* command to activate server*/
    10);/* text string size*/

RegSetValue(hkProtocol,/* handle of protocol key */
    "verb\0", /* name of subkey */
    REG_SZ, /* required */
    "Edit", /* server should edit object */
    4); /* text string size*/

RegCloseKey(hkProtocol);/* closes protocol key and subkeys */
```

### See Also

[RegCloseKey](#), [RegOpenKey](#), [RegSetValue](#)

## RegDeleteKey (3.1)

#include shellapi.h

LONG RegDeleteKey(hkey, lpszSubKey)

HKEY hkey; /\* handle of an open key \*/

LPCSTR lpszSubKey;

\* address of string for subkey to delete

/

The RegDeleteKey function deletes the specified key. When a key is deleted, its value and all of its subkeys are deleted.

Parameter	Description
hkey	Identifies an open key (which can be HKEY_CLASSES_ROOT). The key deleted by the RegDeleteKey function is a subkey of this key.
lpszSubKey	Points to a null-terminated string specifying the subkey to delete. This value should not be NULL.

### Returns

The return value is ERROR\_SUCCESS if the function is successful. Otherwise, it is an error value.

If the error value is ERROR\_ACCESS\_DENIED, either the application does not have delete privileges for the specified key or another application has opened the specified key.

### Example

The following example uses the RegQueryValue function to retrieve the name of an object handler and then calls the RegDeleteKey function to delete the key if its value is nwappobj.dll:

```
char szBuff[80];
LONG cb;
HKEY hkStdFileEditing;

if (RegOpenKey(HKEY_CLASSES_ROOT,
    "NewAppDocument\\protocol\\StdFileEditing",
    &hkStdFileEditing) == ERROR_SUCCESS) {

    cb = sizeof(szBuff);
    if (RegQueryValue(hkStdFileEditing,
        "handler",
        szBuff,
        &cb) == ERROR_SUCCESS
        && lstrcmpi("nwappobj.dll", szBuff) == 0)
        RegDeleteKey(hkStdFileEditing, "handler");
    RegCloseKey(hkStdFileEditing);
}
```

### See Also

RegCloseKey

## RegEnumKey (3.1)

#include shellapi.h

```
LONG RegEnumKey(hkey, iSubkey, lpszBuffer, cbBuffer)
HKEY hkey;          /* handle of key to query          */
DWORD iSubkey;     /* index of subkey to query          */
LPSTR lpszBuffer; /* address of buffer for subkey string */
DWORD cbBuffer;   /* size of subkey buffer          */
```

The RegEnumKey function enumerates the subkeys of a specified key.

Parameter	Description
hkey	Identifies an open key (which can be HKEY_CLASSES_ROOT) for which subkey information is retrieved.
iSubkey	Specifies the index of the subkey to retrieve. This value should be zero for the first call to the RegEnumKey function.
lpszBuffer	Points to a buffer that contains the name of the subkey when the function returns. This function copies only the name of the subkey, not the full key hierarchy, to the buffer.
cbBuffer	Specifies the size, in bytes, of the buffer pointed to by the lpszBuffer parameter.

### Returns

The return value is ERROR\_SUCCESS if the function is successful. Otherwise, it is an error value.

### Comments

The first parameter of the RegEnumKey function must specify an open key. Applications typically precede the call to the RegEnumKey function with a call to the RegOpenKey function and follow it with a call to the RegCloseKey function. Calling RegOpenKey and RegCloseKey is not necessary when the first parameter is HKEY\_CLASSES\_ROOT, because this key is always open and available; however, calling RegOpenKey and RegCloseKey in this case is a time optimization. While an application is using the RegEnumKey function, it should not make calls to any registration functions that might change the key being queried.

To enumerate subkeys, an application should initially set the iSubkey parameter to zero and then increment it on successive calls.

### Example

The following example uses the RegEnumKey function to put the values associated with top-level keys into a list box:

```
HKEY hkRoot;
char szBuff[80], szValue[80];
static DWORD dwIndex;
LONG cb;

if (RegOpenKey(HKEY_CLASSES_ROOT, NULL, &hkRoot) == ERROR_SUCCESS) {
    for (dwIndex = 0; RegEnumKey(hkRoot, dwIndex, szBuff,
        sizeof(szBuff)) == ERROR_SUCCESS; ++dwIndex) {
        if (*szBuff == '.')
            continue;
        cb = sizeof(szValue);
        if (RegQueryValue(hkRoot, (LPSTR) szBuff, szValue,
            &cb) == ERROR_SUCCESS)
            SendDlgItemMessage(hDlg, ID_ENUMLIST, LB_ADDSTRING, 0,
                (LONG) (LPSTR) szValue);
    }
    RegCloseKey(hkRoot);
}
```

See Also



RegQueryValue

## RegOpenKey (3.1)

#include shellapi.h

```
LONG RegOpenKey(hkey, lpszSubKey, lphkResult)
HKEY hkey; /* handle of an open key */
LPCSTR lpszSubKey; /* address of string for subkey to open */
HKEY FAR* lphkResult; /* address of handle of open key */
```

The RegOpenKey function opens the specified key.

Parameter	Description
hkey	Identifies an open key (which can be HKEY_CLASSES_ROOT). The key opened by the RegOpenKey function is a subkey of the key identified by this parameter. This value should not be NULL.
lpszSubKey	Points to a null-terminated string specifying the name of the subkey to open.
lphkResult	Points to the handle of the key that is opened.

### Returns

The return value is ERROR\_SUCCESS if the function is successful. Otherwise, it is an error value.

### Comments

Unlike the [RegCreateKey](#) function, the RegOpenKey function does not create the specified key if the key does not exist in the database.

### Example

The following example uses the RegOpenKey function to retrieve the handle of the StdFileEditing subkey, calls the [RegQueryValue](#) function to retrieve the name of an object handler, and then calls the [RegDeleteKey](#) function to delete the key if its value is nwappobj.dll:

```
char szBuff[80];
LONG cb;
HKEY hkStdFileEditing;

if (RegOpenKey(HKEY_CLASSES_ROOT,
    "NewAppDocument\\protocol\\StdFileEditing",
    &hkStdFileEditing) == ERROR_SUCCESS) {

    cb = sizeof(szBuff);
    if (RegQueryValue(hkStdFileEditing,
        "handler",
        szBuff,
        &cb) == ERROR_SUCCESS
        && lstrcmpi("nwappobj.dll", szBuff) == 0)
        RegDeleteKey(hkStdFileEditing, "handler");
    RegCloseKey(hkStdFileEditing);
}
```

### See Also

[RegCreateKey](#)

## RegQueryValue (3.1)

```
#include shellapi.h
```

```
LONG RegQueryValue(hkey, lpszSubKey, lpszValue, lpcb)
HKEY hkey; /* handle of key to query */
LPCSTR lpszSubKey;
/* address of string for subkey to query
/
LPSTR lpszValue;
* address of buffer for returned string
/
LONG FAR* lpcb;
* address of buffer for size of returned string
/
```

The RegQueryValue function retrieves the text string associated with a specified key.

Parameter	Description
hkey	Identifies a currently open key (which can be HKEY_CLASSES_ROOT). This value should not be NULL.
lpszSubKey	Points to a null-terminated string specifying the name of the subkey of the hkey parameter for which a text string is retrieved. If this parameter is NULL or points to an empty string, the function retrieves the value of the hkey parameter.
lpszValue	Points to a buffer that contains the text string when the function returns.
lpcb	Points to a variable specifying the size, in bytes, of the buffer pointed to by the lpszValue parameter. When the function returns, this variable contains the size of the string copied to lpszValue, including the null-terminating character.

### Returns

The return value is ERROR\_SUCCESS if the function is successful. Otherwise, it is an error value.

### Example

The following example uses the [RegOpenKey](#) function to retrieve the handle of the StdFileEditing subkey, calls the RegQueryValue function to retrieve the name of an object handler and then calls the [RegDeleteKey](#) function to delete the key if its value is nwappobj.dll:

```
char szBuff[80];
LONG cb;
HKEY hkStdFileEditing;

if (RegOpenKey(HKEY_CLASSES_ROOT,
    "NewAppDocument\\protocol\\StdFileEditing",
    &hkStdFileEditing) == ERROR_SUCCESS) {

    cb = sizeof(szBuff);
    if (RegQueryValue(hkStdFileEditing,
        "handler",
        szBuff,
        &cb) == ERROR_SUCCESS
        && lstrcmpi("nwappobj.dll", szBuff) == 0)
        RegDeleteKey(hkStdFileEditing, "handler");
    RegCloseKey(hkStdFileEditing);
}
```

### See Also

[RegEnumKey](#)

## RegSetValue (3.1)

#include shellapi.h

```
LONG RegSetValue(hkey, lpszSubKey, fdwType, lpszValue, cb)
HKEY hkey; /* handle of key */
LPCSTR lpszSubKey; /* address of string for subkey */
DWORD fdwType; /* must be REG_SZ */
LPCSTR lpszValue; /* address of string for key */
DWORD cb; /* ignored */
```

The RegSetValue function associates a text string with a specified key.

Parameter	Description
hkey	Identifies a currently open key (which can be HKEY_CLASSES_ROOT). This value should not be NULL.
lpszSubKey	Points to a null-terminated string specifying the subkey of the hkey parameter with which a text string is associated. If this parameter is NULL or points to an empty string, the function sets the value of the hkey parameter.
fdwType	Specifies the string type. For Windows version 3.1, this value must be REG_SZ.
lpszValue	Points to a null-terminated string specifying the text string to set for the given key.
cb	Specifies the size, in bytes, of the string pointed to by the lpszValue parameter. For Windows version 3.1, this value is ignored.

### Returns

The return value is ERROR\_SUCCESS if the function is successful. Otherwise, it is an error value.

### Comments

If the key specified by the lpszSubKey parameter does not exist, the RegSetValue function creates it.

System performance improves when a call to the RegSetValue function is made for a key that has been opened using the [RegOpenKey](#) function and which will be closed with a call to the [RegCloseKey](#) function.

### Example

The following example uses the RegSetValue function to register a filename extension and its associated class name:

```
RegSetValue(HKEY_CLASSES_ROOT, /* root */
".XXX", /* string for filename extension */
REG_SZ, /* required */
"NewAppDocument", /* class name for extension */
14); /* size of text string */

RegSetValue(HKEY_CLASSES_ROOT, /* root */
"NewAppDocument", /* string for class-definition key */
REG_SZ, /* required */
"New Application", /* text description of class */
15); /* size of text string */
```

### See Also

[RegCloseKey](#), [RegCreateKey](#), [RegOpenKey](#), [RegQueryValue](#)

### Registration functions (3.1)

<u>RegCloseKey</u>	Closes a key and releases the key's handle
<u>RegCreateKey</u>	Creates a specified key
<u>RegDeleteKey</u>	Deletes a specified key
<u>RegEnumKey</u>	Enumerates the subkeys of a specified key
<u>RegOpenKey</u>	Opens a specified key
<u>RegQueryValue</u>	Retrieves the text string for a specified key
<u>RegSetValue</u>	Associates a text string with a specified key

### DefScreenSaverProc (3.1)

```
#include <scrnsave.h>

LRESULT DefScreenSaverProc(HWND hwnd, WPARAM wParam, LPARAM lParam)
/* handle of screen saver window */
UINT msg; /* message */
WPARAM wParam; /* first message parameter */
/
LPARAM lParam; /* second message parameter */
/
```

The DefScreenSaverProc function provides default processing for any messages that a screen saver application does not process. All window messages that are not explicitly processed by the screen saver application's ScreenSaverProc window procedure must be passed to the DefScreenSaverProc function.

Parameter	Description
hwnd	Identifies the screen saver window.
msg	Specifies the message to be processed. The DefScreenSaverProc function responds to messages that affect screen saver operation as detailed in the list in the following "Comments" section. If a screen saver application must perform a different action in response to any of these messages, the application's <u>ScreenSaverProc</u> window procedure should process the message and not call DefScreenSaverProc for that message.
wParam	Specifies 16 bits of additional message-dependent information.
lParam	Specifies 32 bits of additional message-dependent information.

#### Returns

The return value specifies the result of the message processing and depends on the message sent.

#### Comments

A screen saver application's ScreenSaverProc window procedure should use DefScreenSaverProc in place of the DefWindowProc function. The DefScreenSaverProc function passes any messages that do not affect screen saver operation to DefWindowProc.

The DefScreenSaverProc function responds to messages that affect screen saver operation as follows:

Message	Response
WM_ACTIVATE, WM_ACTIVATEAPP, WM_NCACTIVATE	Closes the screen saver if wParam is FALSE, unless the password option is enabled in the configuration dialog box. If the password option is enabled, these messages are ignored. A wParam value of FALSE indicates that the screen saver is losing the input focus. The screen saver is closed by sending a WM_CLOSE message.
WM_SETCURSOR	Removes the cursor from the screen by setting the cursor to NULL.
WM_LBUTTONDOWN, WM_RBUTTONDOWN, WM_MBUTTONDOWN, WM_KEYDOWN, WM_KEYUP, WM_SYSCOMMAND	.2Posts a WM_CLOSE message to close the screen saver window, unless the password option is enabled. If the password option is enabled, a WM_MOUSEMOVE message displays the dialog box created by the <u>DlgGetPassword</u> function. WM_DESTROY Returns FALSE if the wParam parameter of the WM_SYSCOMMAND message is either SC_SCREENSAVE or SC_CLOSE.

#### See Also

ScreenSaverProc

### DlgChangePassword (3.1)

```
#include <scrnsave.h>

BOOL DlgChangePassword(hDlg, message, wParam, lParam)
HWND hDlg;          /* handle of dialog box */
UINT message;      /* message */
/
WPARAM wParam;
/* first message parameter
/
LPARAM lParam;
/* second message parameter
/
```

The DlgChangePassword function receives messages from a dialog box that changes the password for a screen saver.

Parameter	Description
hDlg	Identifies the dialog box that changes the password for a screen saver.
message	Specifies the message.
wParam	Specifies 16 bits of additional message-dependent information.
lParam	Specifies 32 bits of additional message-dependent information.

#### Returns

The return value is nonzero if the function is successful; otherwise, it is zero.

#### Comments

This function is called by the ScreenSaverConfigureDialog function to change the password for a screen saver. An application uses the MakeProcInstance function with DlgChangePassword to display a configuration dialog box.

A password applies to all screen savers using SCRNSAVE.LIB. Whether the password is enabled, however, is specific to a particular screen saver.

The dialog box template for the change password dialog box must use the DLG\_CHANGEPASSWORD identifier (defined as 2000).

The DlgChangePassword function must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

#### See Also

DlgGetPassword, DlgInvalidPassword, ScreenSaverConfigureDialog

## DlgGetPassword (3.1)

```
#include <scrnsave.h>

BOOL DlgGetPassword(hDlg, message, wParam, lParam)
HWND hDlg;          /* handle of dialog box */
UINT message;      /* message */
/
WPARAM wParam;
* first message parameter
/
LPARAM lParam;
* second message parameter
/
```

The DlgGetPassword function receives messages from the dialog box that retrieves the user's password.

Parameter	Description
hDlg	Identifies the dialog box that retrieves the user's password.
message	Specifies the message.
wParam	Specifies 16 bits of additional message-dependent information.
lParam	Specifies 32 bits of additional message-dependent information.

### Returns

The return value is nonzero if the function is successful; otherwise, it is zero.

### Comments

The DlgGetPassword function is provided in SCRNSAVE.LIB. Most applications provide a dialog box template and export the function without explicitly calling it in their code. This reference information for DlgGetPassword is provided for applications that change the default behavior.

The DlgGetPassword function is called by the DefScreenSaverProc function to retrieve the password for a screen saver.

A password applies to all screen savers using SCRNSAVE.LIB. Whether the password is enabled, however, is specific to a particular screen saver.

The dialog box template for the dialog box that retrieves the user's password must use the DLG\_ENTERPASSWORD identifier (defined as 2001).

The DlgGetPassword function must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

### See Also

DefScreenSaverProc, DlgChangePassword, DlgInvalidPassword



### DlgInvalidPassword (3.1)

```
#include <scrnsave.h>

BOOL DlgInvalidPassword(hDlg, message, wParam, lParam)
HWND hDlg;          /* handle of dialog box */
UINT message;      /* message */
/
WPARAM wParam;
* first message parameter
/
LPARAM lParam;
* second message parameter
/
```

The DlgInvalidPassword function displays a dialog box warning that a user's password is invalid.

Parameter	Description
hDlg	Identifies the dialog box that warns that a user's password is invalid.
message	Specifies the message.
wParam	Specifies 16 bits of additional message-dependent information.
lParam	Specifies 32 bits of additional message-dependent information.

#### Returns

The return value is nonzero if the function is successful; otherwise, it is zero.

#### Comments

The DlgInvalidPassword function is provided in SCRNSAVE.LIB. Most applications provide a dialog box template and export the function without explicitly calling it in their code. This reference information for DlgInvalidPassword is provided for applications that change the default behavior.

DlgInvalidPassword is called during processing of the DlgGetPassword function when the user types an incorrect password.

A password applies to all screen savers using SCRNSAVE.LIB. Whether the password is enabled, however, is specific to a particular screen saver.

The dialog box template for the dialog box warning that the user's password is invalid must use the DLG\_INVALIDPASSWORD identifier (defined as 2002).

The DlgInvalidPassword function must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

#### See Also

DlgChangePassword, DlgGetPassword

## HelpMessageFilterHookFunction (3.1)

```
#include <scrnsave.h>
```

```
DWORD HelpMessageFilterHookFunction(nCode, wParam, lParam)
```

```
int nCode; /* identifier of hook */
```

```
WORD wParam; /* virtual-key code */
```

```
LPMMSG lParam;
```

```
/* address of message
```

```
 /
```

The HelpMessageFilterHookFunction function posts a message when the user presses the F1 key while using one of the screen saver dialog boxes.

Parameter	Description
nCode	Specifies a code used by the Windows hook function (also called the message-filter function) to determine how to process the message.
wParam	Specifies the virtual-key code pressed by the user.
lParam	Points to a message identifying the key event.

### Returns

The return value is TRUE if the function posts a message. Otherwise, it specifies the result of the default message processing and is determined by the value of the nCode parameter.

### Comments

The HelpMessageFilterHookFunction function is provided in SCRNSAVE.LIB. Most applications export the function and check for the help message registered by the library without explicitly calling the function in their code. This reference information for HelpMessageFilterHookFunction is provided for applications that change the default behavior.

The HelpMessageFilterHookFunction function posts a registered message called MyHelpMessage. An application should check for this message in its ScreenSaverConfigureDialog function.

The HelpMessageFilterHookFunction function must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

### See Also

ScreenSaverConfigureDialog

## RegisterDialogClasses (3.1)

```
#include <scrnsave.h>
```

```
BOOL RegisterDialogClasses(hInst)  
HANDLE hInst;          /* handle of application instance */
```

The RegisterDialogClasses function registers any special or nonstandard window classes needed by a screen saver application's configuration dialog box.

Parameter	Description
hInst	Identifies an instance of the module that is registering the window classes.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The RegisterDialogClasses function should not be exported. It is called by routines defined in the SCRNSAVE.LIB file.

If a screen saver does not register any special window classes for the configuration dialog box, the RegisterDialogClasses function can simply return a nonzero value.

### See Also

[ScreenSaverConfigureDialog](#)

## ScreenSaverConfigureDialog (3.1)

```
#include <scrnsave.h>

BOOL ScreenSaverConfigureDialog(hdlg, wmsg, wParam, lParam)
HWND hdlg;          /* handle of dialog box */
UINT wmsg;          /* message */
WPARAM wParam;
/* first message parameter
/
LPARAM lParam;
/* second message parameter
/
```

The ScreenSaverConfigureDialog function receives messages sent to a screen saver application's configuration dialog box. A screen saver application that supports user configuration must provide this function.

Parameter	Description
hdlg	Identifies the configuration dialog box.
wmsg	Specifies the message.
wParam	Specifies 16 bits of additional message-dependent information.
lParam	Specifies 32 bits of additional message-dependent information.

### Returns

The return value is nonzero if the function processes the message or zero if it does not, except in response to a WM\_INITDIALOG message. In response to a WM\_INITDIALOG message, ScreenSaverConfigureDialog should return zero if it calls the SetFocus function to set the input focus to one of the controls in the dialog box. Otherwise, it should return nonzero, in which case the system sets the input focus to the first control in the dialog box that can be given the focus.

### Comments

An application uses the MakeProcInstance function with ScreenSaverConfigureDialog to display a configuration dialog box.

The dialog box template for the configuration dialog box must have the DLG\_SCRNSAVECONFIGURE identifier.

A screen saver application should save its configuration settings in the CONTROL.INI file.

The dialog box procedure is used only if the default window class (WC\_DIALOG) is used for the dialog box. The default class is used if no explicit class is given in the dialog box template. Although the dialog box procedure is similar to a window procedure, it must not call the DefWindowProc function to process unwanted messages. Unwanted messages are processed internally by the default dialog box procedure.

The ScreenSaverConfigureDialog function must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

### See Also

MakeProcInstance, RegisterDialogClasses

### ScreenSaverProc (3.1)

```
#include <scrnsave.h>
```

```
LRESULT ScreenSaverProc(hwnd, wmsg, wParam, lParam)
```

```
HWND hwnd; /* handle of screen saver window */
```

```
unsigned wmsg; /* message */
```

```
/
```

```
UINT wParam;
```

```
* first message parameter
```

```
/
```

```
LPARAM lParam;
```

```
* second message parameter
```

```
/
```

The ScreenSaverProc function receives messages sent to a screen saver window.

Parameter	Description
hwnd	Identifies the window.
wmsg	Specifies the message.
wParam	Specifies 16 bits of additional message-dependent information.
lParam	Specifies 32 bits of additional message-dependent information.

#### Returns

The return value is the result of the message processing. It depends on the message that is processed.

#### Comments

A screen saver application's ScreenSaverProc window procedure should use the DefScreenSaverProc function instead of the DefWindowProc function to provide default message processing. The DefScreenSaverProc function passes any messages that do not affect screen saver operations to DefWindowProc.

The ScreenSaverProc function must be exported by including it in an EXPORTS statement in the application's module-definition (.DEF) file.

#### See Also

DefScreenSaverProc

## Screen Saver functions

DefScreenSaverProc

Calls default screen saver window procedure

DlgChangePassword

Changes the password for a screen saver

DlgGetPassword

Retrieves the password for a screen saver

DlgInvalidPassword

Warns of an invalid screen saver password

HelpMessageFilterHookFunction

Posts a screen saver help message

RegisterDialogClasses

Registers screen-saver dialog box classes

ScreenSaverConfigureDialog

Processes config. dialog box messages

ScreenSaverProc

Processes screen saver window messages

### ExtractIcon (3.1)

```
#include shellapi.h
```

```
HICON ExtractIcon(hinst, lpszExeName, iIcon)
```

```
HINSTANCE hinst; /* instance handle */
```

```
LPCSTR lpszExeName; /* address of string for file */
```

```
/* address of string for file */
```

```
/
```

```
UINT iIcon;
```

```
/* index of icon to retrieve */
```

```
/
```

The ExtractIcon function retrieves the handle of an icon from a specified executable file, dynamic-link library (DLL), or icon file.

Parameter	Description
hinst	Identifies the instance of the application calling the function.
lpszExeName	Points to a null-terminated string specifying the name of an executable file, dynamic-link library, or icon file.
iIcon	Specifies the index of the icon to be retrieved. If this parameter is zero, the function returns the handle of the first icon in the specified file. If the parameter is -1, the function returns the total number of icons in the specified file.

#### Returns

The return value is the handle of an icon if the function is successful. It is 1 if the file specified in the lpszExeName parameter is not an executable file, dynamic-link library, or icon file. Otherwise, it is NULL, indicating that the file contains no icons.

### FindExecutable (3.1)

```
#include shellapi.h
```

```
HINSTANCE FindExecutable(lpszFile, lpszDir, lpszResult)
```

```
LPCSTR lpszFile; /* address of string for filename */
```

```
LPCSTR lpszDir;
```

```
/* address of string for default directory
```

```
/
```

```
LPSTR lpszResult;
```

```
/* address of string for executable file on return
```

```
/
```

The FindExecutable function finds and retrieves the executable filename that is associated with a specified filename.

Parameter	Description
lpszFile	Points to a null-terminated string specifying a filename. This can be a document or executable file.
lpszDir	Points to a null-terminated string specifying the drive letter and path for the default directory.
lpszResult	Points to a buffer that receives the name of an executable file when the function returns. This null-terminated string specifies the application that is started when the Open command is chosen from the File menu in File Manager.

#### Returns

The return value is greater than 32 if the function is successful. If the return value is less than or equal to 32, it specifies an error code.

#### Errors

The FindExecutable function returns 31 if there is no association for the specified file type. The other possible error values are as follows:

Value	Meaning
0	System was out of memory, executable file was corrupt, or relocations were invalid.
2	File was not found.
3	Path was not found.
5	Attempt was made to dynamically link to a task, or there was a sharing or network-protection error.
6	Library required separate data segments for each task.
8	There was insufficient memory to start the application.
10	Windows version was incorrect.
11	Executable file was invalid. Either it was not a Windows application or there was an error in the .EXE image.
12	Application was designed for a different operating system.
13	Application was designed for MS-DOS 4.0.
14	Type of executable file was unknown.
15	Attempt was made to load a real-mode application (developed for an earlier version of Windows).
16	Attempt was made to load a second instance of an executable file containing multiple data segments that were not marked read-only.
19	Attempt was made to load a compressed executable file. The file must be decompressed before it can be loaded.
20	Dynamic-link library (DLL) file was invalid. One of the DLLs required to run this application was corrupt.
21	Application requires Microsoft Windows 32-bit extensions.

#### Comments

The filename specified in the lpszFile parameter is associated with an executable file when an association has been registered between that file's filename extension and an executable file in the registration database. An application that produces files with a given filename extension typically associates the extension with an executable file when the application is installed.



See Also  
RegQueryValue, ShellExecute

### ShellExecute (3.1)

```
#include shellapi.h

HINSTANCE ShellExecute(hwnd, lpszOp, lpszFile, lpszParams, lpszDir, fsShowCmd)
HWND hwnd; /* handle of parent window */
LPCSTR lpszOp;
/* address of string for operation to perform
/
LPCSTR lpszFile;
/* address of string for filename
/
LPCSTR lpszParams;
/* address of string for executable-file parameters
/
LPCSTR lpszDir;
/* address of string for default directory
/
int fsShowCmd;
/* whether file is shown when opened
/
```

The ShellExecute function opens or prints the specified file.

Parameter	Description
hwnd	Identifies the parent window. This window receives any message boxes an application produces (for example, for error reporting).
lpszOp	Points to a null-terminated string specifying the operation to perform. This string can be "open" or "print". If this parameter is NULL, "open" is the default value.
lpszFile	Points to a null-terminated string specifying the file to open.
lpszParams	Points to a null-terminated string specifying parameters passed to the application when the lpszFile parameter specifies an executable file. If lpszFile points to a string specifying a document file, this parameter is NULL.
lpszDir	Points to a null-terminated string specifying the default directory.
fsShowCmd	Specifies whether the application window is to be shown when the application is opened. This parameter can be one of the following values:

Value	Meaning
SW_HIDE	Hides the window and passes activation to another window.
SW_MINIMIZE	Minimizes the specified window and activates the top-level window in the system's list.
SW_RESTORE	Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_SHOWNORMAL).
SW_SHOW	Activates a window and displays it in its current size and position.
SW_SHOWMAXIMIZED	Activates a window and displays it as a maximized window.
SW_SHOWMINIMIZED	Activates a window and displays it as an icon.
SW_SHOWMINNOACTIVE	Displays a window as an icon. The window that is currently active remains active.
SW_SHOWNA	Displays a window in its current state. The window that is currently active remains active.
SW_SHOWNOACTIVATE	Displays a window in its most recent size and position. The window that is currently active remains active.
SW_SHOWNORMAL	Activates and displays a window. If the window is minimized or maximized, Windows restores it to

its original size and position (same as SW\_RESTORE).

#### Returns

The return value is the instance handle of the application that was opened or printed, if the function is successful. (This handle could also be the handle of a DDE server application.) A return value less than or equal to 32 specifies an error. The possible error values are listed in the following Comments section.

#### Errors

The ShellExecute function returns the value 31 if there is no association for the specified file type or if there is no association for the specified action within the file type. The other possible error values are as follows:

Value	Meaning
0	System was out of memory, executable file was corrupt, or relocations were invalid.
2	File was not found.
3	Path was not found.
5	Attempt was made to dynamically link to a task, or there was a sharing or network-protection error.
6	Library required separate data segments for each task.
8	There was insufficient memory to start the application.
10	Windows version was incorrect.
11	Executable file was invalid. Either it was not a Windows application or there was an error in the .EXE image.
12	Application was designed for a different operating system.
13	Application was designed for MS-DOS 4.0.
14	Type of executable file was unknown.
15	Attempt was made to load a real-mode application (developed for an earlier version of Windows).
16	Attempt was made to load a second instance of an executable file containing multiple data segments that were not marked read-only.
19	Attempt was made to load a compressed executable file. The file must be decompressed before it can be loaded.
20	Dynamic-link library (DLL) file was invalid. One of the DLLs required to run this application was corrupt.
21	Application requires Microsoft Windows 32-bit extensions.

#### Comments

The file specified by the `lpszFile` parameter can be a document file or an executable file. If it is a document file, this function opens or prints it, depending on the value of the `lpszOp` parameter. If it is an executable file, this function opens it, even if the string "print" is pointed to by `lpszOp`.

#### See Also

[FindExecutable](#)

### Shell functions (3.1)

<u>ExtractIcon</u>	Retrieves the handle of an icon from an executable file
<u>FindExecutable</u>	Retrieves executable filename for a specified file
<u>ShellExecute</u>	Opens or prints the specified file

## AllocDiskSpace (3.1)

```
#include stress.h
int AllocDiskSpace(lLeft, uDrive)
long lLeft;          /* number of bytes left available */
UINT uDrive;        /* disk partition */
```

The AllocDiskSpace function creates a file that is large enough to ensure that the specified amount of space or less is available on the specified disk partition. The file, called STRESS.EAT, is created in the root directory of the disk partition.

If STRESS.EAT already exists when AllocDiskSpace is called, the function deletes it and creates a new one.

Parameter	Description								
lLeft	Specifies the number of bytes to leave available on the disk.								
uDrive	Specifies the disk partition on which to create the STRESS.EAT file. This parameter must be one of the following values:								
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>EDS_WIN</td><td>Creates the file on the Windows partition.</td></tr><tr><td>EDS_CUR</td><td>Creates the file on the current partition.</td></tr><tr><td>EDS_TEMP</td><td>Creates the file on the partition that contains the TEMP directory.</td></tr></tbody></table>	Value	Meaning	EDS_WIN	Creates the file on the Windows partition.	EDS_CUR	Creates the file on the current partition.	EDS_TEMP	Creates the file on the partition that contains the TEMP directory.
Value	Meaning								
EDS_WIN	Creates the file on the Windows partition.								
EDS_CUR	Creates the file on the current partition.								
EDS_TEMP	Creates the file on the partition that contains the TEMP directory.								

### Returns

The return value is greater than zero if the function is successful; it is zero if the function could not create a file; or it is -1 if at least one of the parameters is invalid.

### Comments

In two situations, the amount of free space left on the disk may be less than the number of bytes specified in the lLeft parameter: when the amount of free space on the disk is less than the number in lLeft when an application calls AllocDiskSpace, or when the value of lLeft is not an exact multiple of the disk cluster size.

The UnAllocDiskSpace function deletes the file created by AllocDiskSpace.

See Also

UnAllocDiskSpace

## AllocFileHandles (3.1)

#include stress.h

```
int AllocFileHandles(Left)
```

```
int Left; /* number of file handles to leave available */
```

The AllocFileHandles function allocates file handles until only the specified number of file handles is available to the current instance of the application. If this or a smaller number of handles is available when an application calls AllocFileHandles, the function returns immediately.

Before allocating new handles, this function frees any handles previously allocated by AllocFileHandles.

Parameter	Description
Left	Specifies the number of file handles to leave available.

### Returns

The return value is greater than zero if AllocFileHandles successfully allocates at least one file handle. The return value is zero if fewer than the specified number of file handles were available when the application called AllocFileHandles. The return value is -1 if the Left parameter is negative.

### Comments

AllocFileHandles will not allocate more than 256 file handles, regardless of the number available to the application.

The UnAllocFileHandles function frees all file handles previously allocated by AllocFileHandles.

### See Also

UnAllocFileHandles

## AllocGDIMem (3.1)

#include stress.h

```
BOOL AllocGDIMem(uLeft)
```

```
UINT uLeft; /* number of bytes to leave available */
```

The AllocGDIMem function allocates memory in the graphics device interface (GDI) heap until only the specified number of bytes is available. Before making any new memory allocations, this function frees memory previously allocated by AllocGDIMem.

Parameter	Description
-----------	-------------

uLeft	Specifies the amount of memory, in bytes, to leave available in the GDI heap.
-------	---

Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

Comments

The FreeAllGDIMem function frees all memory allocated by AllocGDIMem.

See Also

FreeAllGDIMem

## AllocMem (3.1)

#include stress.h

BOOL AllocMem(dwLeft)

DWORD dwLeft; /\*smallest memory allocation \*/

The AllocMem function allocates global memory until only the specified number of bytes is available in the global heap. Before making any new memory allocations, this function frees memory previously allocated by AllocMem.

Parameter	Description
dwLeft	Specifies the smallest size, in bytes, of memory allocations to make.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The FreeAllMem function frees all memory allocated by AllocMem.

See Also

FreeAllMem



## AllocUserMem (3.1)

#include stress.h

BOOL AllocUserMem(uContig)

UINT uContig; /\* smallest memory allocation \*/

The AllocUserMem function allocates memory in the USER heap until only the specified number of bytes is available. Before making any new allocations, this function frees memory previously allocated by AllocUserMem.

Parameter	Description
uContig	Specifies the smallest size, in bytes, of memory allocations to make.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The FreeAllUserMem function frees all memory allocated by AllocUserMem.

See Also

FreeAllUserMem

## FreeAllGDIMem (3.1)

#include stress.h

void FreeAllGDIMem(void)

The FreeAllGDIMem function frees all memory allocated by the AllocGDIMem function.

Returns

This function does not return a value.

See Also

AllocGDIMem

## FreeAllMem (3.1)

#include stress.h

void FreeAllMem(void)

The FreeAllMem function frees all memory allocated by the AllocMem function.

Returns

This function does not return a value.

See Also

AllocMem

## FreeAllUserMem (3.1)

#include stress.h

void FreeAllUserMem(void)

The FreeAllUserMem function frees all memory allocated by the AllocUserMem function.

Returns

This function does not return a value.

See Also

AllocUserMem

### GetFreeFileHandles (3.1)

```
#include stress.h
```

```
int GetFreeFileHandles(void)
```

The GetFreeFileHandles function returns the number of file handles available to the current instance.

Returns

The return value is the number of file handles available to the current instance.

## UnAllocDiskSpace (3.1)

```
#include stress.h
```

```
void UnAllocDiskSpace(drive)
UINT drive;
```

The UnAllocDiskSpace function deletes the STRESS.EAT file from the root directory of the specified drive. This frees the disk space previously consumed by the [AllocDiskSpace](#) function.

Parameter	Description								
drive	Specifies the disk partition on which to delete the STRESS.EAT file. This can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>EDS_WIN</td><td>Deletes the file on the Windows partition.</td></tr><tr><td>EDS_CUR</td><td>Deletes the file on the current partition.</td></tr><tr><td>EDS_TEMP</td><td>Deletes the file on the partition that contains the TEMP directory.</td></tr></tbody></table>	Value	Meaning	EDS_WIN	Deletes the file on the Windows partition.	EDS_CUR	Deletes the file on the current partition.	EDS_TEMP	Deletes the file on the partition that contains the TEMP directory.
Value	Meaning								
EDS_WIN	Deletes the file on the Windows partition.								
EDS_CUR	Deletes the file on the current partition.								
EDS_TEMP	Deletes the file on the partition that contains the TEMP directory.								

### Returns

This function does not return a value.

### See Also

[AllocDiskSpace](#)

## UnAllocFileHandles (3.1)

#include stress.h

void UnAllocFileHandles(void)

The UnAllocFileHandles function frees all file handles allocated by the AllocFileHandles function.

Returns

This function does not return a value.

See Also

AllocFileHandles

### Stress Functions (3.1)

<u>AllocDiskSpace</u>	Creates a file to consume space on a disk partition
<u>AllocFileHandles</u>	Allocates up to 256 file handles
<u>AllocGDIMem</u>	Allocates memory in the GDI heap
<u>AllocMem</u>	Allocates global memory
<u>AllocUserMem</u>	Allocates memory in the USER heap
<u>FreeAllGDIMem</u>	Frees all memory allocated by the AllocGDIMem function
<u>FreeAllMem</u>	Frees all memory allocated by the AllocMem function
<u>FreeAllUserMem</u>	Frees all memory allocated by the AllocUserMem function
<u>GetFreeFileHandles</u>	Returns the number of free file handles
<u>UnAllocDiskSpace</u>	Deletes file created by AllocDiskSpace and frees space
<u>UnAllocFileHandles</u>	Frees file handles allocated by AllocFileHandles



## ABC (3.1)

```
typedef struct tagABC {    /* abc */
    int    abcA;
    UINT   abcB;
    int    abcC;
} ABC;
```

The ABC structure contains the width of a character in a TrueType font.

Member	Description
abcA	Specifies the "A" spacing of the character. A spacing is the distance to add to the current position before drawing the character glyph.
abcB	Specifies the "B" spacing of the character. B spacing is the width of the drawn portion of the character glyph.
abcC	Specifies the "C" spacing of the character. C spacing is the distance to add to the current position to provide white space to the right of the character glyph.

### Comments

The total width of a character is the sum of the A, B, and C spaces. Either the A or the C space can be negative, to indicate underhangs or overhangs.

### See Also

[GetCharABCWidths](#)

## BITMAP (2.x)

```
typedef struct tagBITMAP { /* bm */
    int bmType;
    int bmWidth;
    int bmHeight;
    int bmWidthBytes;
    BYTE    bmPlanes;
    BYTE    bmBitsPixel;
    void FAR* bmBits;
} BITMAP;
```

The BITMAP structure defines the height, width, color format, and bit values of a logical bitmap.

Member	Description
bmType	Specifies the bitmap type. For logical bitmaps, this member must be zero.
bmWidth	Specifies the width of the bitmap, in pixels. The width must be greater than zero.
bmHeight	Specifies the height of the bitmap, in raster lines. The height must be greater than zero.
bmWidthBytes	Specifies the number of bytes in each raster line. This value must be an even number since graphics device interface (GDI) assumes that the bit values of a bitmap form an array of integer (two-byte) values. In other words, bmWidthBytes * 8 must be the next multiple of 16 greater than or equal to the value obtained when the bmWidth member is multiplied by the bmBitsPixel member.
bmPlanes	Specifies the number of color planes in the bitmap.
bmBitsPixel	Specifies the number of adjacent color bits on each plane needed to define a pixel.
bmBits	Points to the location of the bit values for the bitmap. The bmBits member must be a long pointer to an array of one-byte values.

### Comments

The currently used bitmap formats are monochrome and color. The monochrome bitmap uses a one-bit, one-plane format. Each scan is a multiple of 16 bits.

Scans are organized as follows for a monochrome bitmap of height n:

```
Scan 0
Scan 1
.
.
.
Scan n-2
Scan n-1
```

The pixels on a monochrome device are either black or white. If the corresponding bit in the bitmap is 1, the pixel is turned on (white). If the corresponding bit in the bitmap is zero, the pixel is turned off (black).

All devices support bitmaps that have the RC\_BITBLT bit set in the RASTERCAPS index of the GetDeviceCaps function.

Each device has its own unique color format. In order to transfer a bitmap from one device to another, use the GetDIBits and SetDIBits functions.

### See Also

CreateBitmapIndirect, GetDIBits, GetObject, SetDIBits

## BITMAPCOREHEADER (3.0)

```
typedef struct tagBITMAPCOREHEADER {      /* bmch */
    DWORD   bcSize;
    short   bcWidth;
    short   bcHeight;
    WORD    bcPlanes;
    WORD    bcBitCount;
} BITMAPCOREHEADER;
```

The BITMAPCOREHEADER structure contains information about the dimensions and color format of a device-independent bitmap (DIB). Windows applications should use the BITMAPINFOHEADER structure instead of BITMAPCOREHEADER whenever possible.

Member	Description
bcSize	Specifies the number of bytes required by the BITMAPCOREHEADER structure.
bcWidth	Specifies the width of the bitmap, in pixels.
bcHeight	Specifies the height of the bitmap, in pixels.
bcPlanes	Specifies the number of planes for the target device. This member must be set to 1.
bcBitCount	Specifies the number of bits per pixel. This value must be 1, 4, 8, or 24.

### Comments

The BITMAPCOREINFO structure combines the BITMAPCOREHEADER structure and a color table to provide a complete definition of the dimensions and colors of a DIB. See the description of the BITMAPCOREINFO structure for more information about specifying a DIB.

An application should use the information stored in the bcSize member to locate the color table in a BITMAPCOREINFO structure with a method such as the following:

```
lpColor = ((LPSTR) pBitmapCoreInfo + (UINT) (pBitmapCoreInfo->bcSize))
```

See Also

BITMAPCOREINFO, BITMAPINFOHEADER, BITMAPINFOHEADER

## BITMAPCOREINFO (3.0)

```
typedef struct tagBITMAPCOREINFO { /* bmci */
    BITMAPCOREHEADER bmciHeader;
    RGBTRIPLE    bmciColors[1];
} BITMAPCOREINFO;
```

The BITMAPCOREINFO structure fully defines the dimensions and color information for a device-independent bitmap (DIB). Windows applications should use the BITMAPINFO structure instead of BITMAPCOREINFO whenever possible.

Member	Description
bmciHeader	Specifies a <u>BITMAPCOREHEADER</u> structure that contains information about the dimensions and color format of a DIB.
bmciColors	Specifies an array of <u>RGBTRIPLE</u> structures that define the colors in the bitmap.

### Comments

The BITMAPCOREINFO structure describes the dimensions and colors of a bitmap. It is followed immediately in memory by an array of bytes which define the pixels of the bitmap. The bits in the array are packed together, but each scan line must be zero-padded to end on a LONG boundary. Segment boundaries, however, can appear anywhere in the bitmap. The origin of the bitmap is the lower-left corner.

The bcBitCount member of the BITMAPCOREHEADER structure determines the number of bits that define each pixel and the maximum number of colors in the bitmap. This member may be set to any of the following values:

Value	Meaning
1	The bitmap is monochrome, and the bmciColors member must contain two entries. Each bit in the bitmap array represents a pixel. If the bit is clear, the pixel is displayed with the color of the first entry in the bmciColors table. If the bit is set, the pixel has the color of the second entry in the table.
4	The bitmap has a maximum of 16 colors, and the bmciColors member contains 16 entries. Each pixel in the bitmap is represented by a four-bit index into the color table. For example, if the first byte in the bitmap is 0x1F, the byte represents two pixels. The first pixel contains the color in the second table entry, and the second pixel contains the color in the sixteenth table entry.
8	The bitmap has a maximum of 256 colors, and the bmciColors member contains 256 entries. In this case, each byte in the array represents a single pixel.
24	The bitmap has a maximum of 2 <sup>24</sup> colors. The bmciColors member is NULL, and each 3-byte sequence in the bitmap array represents the relative intensities of red, green, and blue, respectively, of a pixel.

The colors in the bmciColors table should appear in order of importance.

Alternatively, for functions that use DIBs, the bmciColors member can be an array of 16-bit unsigned integers that specify an index into the currently realized logical palette instead of explicit RGB values. In this case, an application using the bitmap must call DIB functions with the wUsage parameter set to DIB\_PAL\_COLORS.

Note: The bmciColors member should not contain palette indexes if the bitmap is to be stored in a file or transferred to another application. Unless the application uses the bitmap exclusively and under its complete control, the bitmap color table should contain explicit RGB values.

### See Also

BITMAPINFO, BITMAPCOREHEADER, RGBTRIPLE

■ **BITMAPFILEHEADER (3.0)**

```
typedef struct tagBITMAPFILEHEADER {      /* bmfh */
    UINT    bfType;
    DWORD   bfSize;
    UINT    bfReserved1;
    UINT    bfReserved2;
    DWORD   bfOffBits;
} BITMAPFILEHEADER;
```

The BITMAPFILEHEADER structure contains information about the type, size, and layout of a device-independent bitmap (DIB) file.

Member	Description
bfType	Specifies the type of file. This member must be BM.
bfSize	Specifies the size of the file, in bytes.
bfReserved1	Reserved; must be set to zero.
bfReserved2	Reserved; must be set to zero.
bfOffBits	Specifies the byte offset from the BITMAPFILEHEADER structure to the actual bitmap data in the file.

Comments

A BITMAPINFO or BITMAPCOREINFO structure immediately follows the BITMAPFILEHEADER structure in the DIB file.

See Also

BITMAPCOREINFO, BITMAPINFO

## Corrections

The previous documentation stated that the bfSize member specifies the size of the file in DWORDS. This was wrong; bfSize specifies the size of the file in bytes.

## BITMAPINFO (3.0)

```
typedef struct tagBITMAPINFO { /* bmi */
    BITMAPINFOHEADER bmiHeader;
    RGBQUAD bmiColors[1];
} BITMAPINFO;
```

The BITMAPINFO structure fully defines the dimensions and color information for a Windows 3.0 or later device-independent bitmap (DIB).

Member	Description
bmiHeader	Specifies a <u>BITMAPINFOHEADER</u> structure that contains information about the dimensions and color format of a DIB.
bmiColors	Specifies an array of <u>RGBQUAD</u> structures that define the colors in the bitmap.

### Comments

A Windows 3.0 or later DIB consists of two distinct parts: a BITMAPINFO structure, which describes the dimensions and colors of the bitmap, and an array of bytes defining the pixels of the bitmap. The bits in the array are packed together, but each scan line must be zero-padded to end on a LONG boundary. Segment boundaries, however, can appear anywhere in the bitmap. The origin of the bitmap is the lower-left corner.

The biBitCount member of the BITMAPINFOHEADER structure determines the number of bits which define each pixel and the maximum number of colors in the bitmap. This member may be set to any of the following values:

Value	Meaning
1	The bitmap is monochrome, and the bmiColors member must contain two entries. Each bit in the bitmap array represents a pixel. If the bit is clear, the pixel is displayed with the color of the first entry in the bmiColors table. If the bit is set, the pixel has the color of the second entry in the table.
4	The bitmap has a maximum of 16 colors, and the bmiColors member contains 16 entries. Each pixel in the bitmap is represented by a four-bit index into the color table. For example, if the first byte in the bitmap is 0x1F, the byte represents two pixels. The first pixel contains the color in the second table entry, and the second pixel contains the color in the sixteenth table entry.
8	The bitmap has a maximum of 256 colors, and the bmiColors member contains 256 entries. In this case, each byte in the array represents a single pixel.
24	The bitmap has a maximum of 2 <sup>24</sup> colors. The bmiColors member is NULL, and each 3-byte sequence in the bitmap array represents the relative intensities of red, green, and blue, respectively, of a pixel.

The biClrUsed member of the BITMAPINFOHEADER structure specifies the number of color indexes in the color table actually used by the bitmap. If the biClrUsed member is set to zero, the bitmap uses the maximum number of colors corresponding to the value of the biBitCount member.

The colors in the bmiColors table should appear in order of importance.

Alternatively, for functions that use DIBs, the bmiColors member can be an array of 16-bit unsigned integers that specify an index into the currently realized logical palette instead of explicit RGB values. In this case, an application using the bitmap must call DIB functions with the wUsage parameter set to DIB\_PAL\_COLORS.

Note: The bmiColors member should not contain palette indexes if the bitmap is to be stored in a file or transferred to another application. Unless the application uses the bitmap exclusively and under its complete control, the bitmap color table should contain explicit RGB values.

See Also

BITMAPINFOHEADER, RGBQUAD

## BITMAPINFOHEADER (3.0)

```
typedef struct tagBITMAPINFOHEADER {      /* bmih */
    DWORD   biSize;
    LONG    biWidth;
    LONG    biHeight;
    WORD    biPlanes;
    WORD    biBitCount;
    DWORD   biCompression;
    DWORD   biSizeImage;
    LONG    biXPelsPerMeter;
    LONG    biYPelsPerMeter;
    DWORD   biClrUsed;
    DWORD   biClrImportant;
} BITMAPINFOHEADER;
```

The BITMAPINFOHEADER structure contains information about the dimensions and color format of a Windows 3.0 or later device-independent bitmap (DIB).

Member	Description								
biSize	Specifies the number of bytes required by the BITMAPINFOHEADER structure.								
biWidth	Specifies the width of the bitmap, in pixels.								
biHeight	Specifies the height of the bitmap, in pixels.								
biPlanes	Specifies the number of planes for the target device. This member must be set to 1.								
biBitCount	Specifies the number of bits per pixel. This value must be 1, 4, 8, or 24.								
biCompression	Specifies the type of compression for a compressed bitmap. It can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>BI_RGB</td><td>Specifies that the bitmap is not compressed.</td></tr><tr><td>BI_RLE8</td><td>Specifies a run-length encoded format for bitmaps with 8 bits per pixel. The compression format is a 2-byte format consisting of a count byte followed by a byte containing a color index. For more information, see the following Comments section.</td></tr><tr><td>BI_RLE4</td><td>Specifies a run-length encoded format for bitmaps with 4 bits per pixel. The compression format is a 2-byte format consisting of a count byte followed by two word-length color indexes. For more information, see the following Comments section.</td></tr></tbody></table>	Value	Meaning	BI_RGB	Specifies that the bitmap is not compressed.	BI_RLE8	Specifies a run-length encoded format for bitmaps with 8 bits per pixel. The compression format is a 2-byte format consisting of a count byte followed by a byte containing a color index. For more information, see the following Comments section.	BI_RLE4	Specifies a run-length encoded format for bitmaps with 4 bits per pixel. The compression format is a 2-byte format consisting of a count byte followed by two word-length color indexes. For more information, see the following Comments section.
Value	Meaning								
BI_RGB	Specifies that the bitmap is not compressed.								
BI_RLE8	Specifies a run-length encoded format for bitmaps with 8 bits per pixel. The compression format is a 2-byte format consisting of a count byte followed by a byte containing a color index. For more information, see the following Comments section.								
BI_RLE4	Specifies a run-length encoded format for bitmaps with 4 bits per pixel. The compression format is a 2-byte format consisting of a count byte followed by two word-length color indexes. For more information, see the following Comments section.								
biSizeImage	Specifies the size, in bytes, of the image. It is valid to set this member to zero if the bitmap is in the BI_RGB format.								
biXPelsPerMeter	Specifies the horizontal resolution, in pixels per meter, of the target device for the bitmap. An application can use this value to select a bitmap from a resource group that best matches the characteristics of the current device.								
biYPelsPerMeter	Specifies the vertical resolution, in pixels per meter, of the target device for the bitmap.								
biClrUsed	<p>Specifies the number of color indexes in the color table actually used by the bitmap. If this value is zero, the bitmap uses the maximum number of colors corresponding to the value of the biBitCount member. For more information on the maximum sizes of the color table, see the description of the <a href="#">BITMAPINFO</a> structure earlier in this topic.</p> <p>If the biClrUsed member is nonzero, it specifies the actual number of colors that the graphics engine or device driver will access if the biBitCount member is less than 24. If biBitCount is set to 24, biClrUsed specifies the size of the reference color table used to optimize performance of Windows color palettes.</p> <p>If the bitmap is a packed bitmap (that is, a bitmap in which the bitmap array immediately follows the BITMAPINFO header and which is referenced by a single pointer), the biClrUsed member must be set to zero or to the actual size of the color table.</p>								
biClrImportant	Specifies the number of color indexes that are considered important for displaying the bitmap. If this value is zero, all colors are important.								



## Comments

The BITMAPINFO structure combines the BITMAPINFOHEADER structure and a color table to provide a complete definition of the dimensions and colors of a Windows 3.0 or later DIB. For more information about specifying a Windows 3.0 DIB, see the description of the BITMAPINFO structure.

An application should use the information stored in the biSize member to locate the color table in a BITMAPINFO structure as follows:

```
pColor = ((LPSTR) pBitmapInfo + (WORD) (pBitmapInfo->bmiHeader.biSize)
)
```

Windows supports formats for compressing bitmaps that define their colors with 8 bits per pixel and with 4 bits per pixel. Compression reduces the disk and memory storage required for the bitmap. The following paragraphs describe these formats.

### BI\_RLE8

When the biCompression member is set to BI\_RLE8, the bitmap is compressed using a run-length encoding format for an 8-bit bitmap. This format may be compressed in either of two modes: encoded and absolute. Both modes can occur anywhere throughout a single bitmap.

Encoded mode consists of two bytes: the first byte specifies the number of consecutive pixels to be drawn using the color index contained in the second byte. In addition, the first byte of the pair can be set to zero to indicate an escape that denotes an end of line, end of bitmap, or a delta. The interpretation of the escape depends on the value of the second byte of the pair. The following list shows the meaning of the second byte:

Value	Meaning
0	End of line.
1	End of bitmap.
2	Delta. The two bytes following the escape contain unsigned values indicating the horizontal and vertical offset of the next pixel from the current position.

Absolute mode is signaled by the first byte set to zero and the second byte set to a value between 0x03 and 0xFF. In absolute mode, the second byte represents the number of bytes that follow, each of which contains the color index of a single pixel. When the second byte is set to 2 or less, the escape has the same meaning as in encoded mode. In absolute mode, each run must be aligned on a word boundary.

The following example shows the hexadecimal values of an 8-bit compressed bitmap:

```
03 04 05 06 00 03 45 56 67 00 02 78 00 02 05 01
02 78 00 00 09 1E 00 01
```

This bitmap would expand as follows (two-digit values represent a color index for a single pixel):

```
04 04 04
06 06 06 06 06
45 56 67
78 78
move current position 5 right and 1 down
78 78
end of line
1E 1E 1E 1E 1E 1E 1E 1E 1E
end of RLE bitmap
```

### BI\_RLE4

When the biCompression member is set to BI\_RLE4, the bitmap is compressed using a run-length encoding (RLE) format for a 4-bit bitmap, which also uses encoded and absolute modes. In encoded mode, the first byte of the pair contains the number of pixels to be drawn using the color indexes in the second byte. The second byte contains two color indexes, one in its high-order nibble (that is, its low-order four bits) and one in its low-order nibble. The first of the pixels is drawn using the color specified by the high-order nibble, the second is drawn using the color in the low-order nibble, the third is drawn with the color in the high-order nibble, and so on, until all the pixels specified by the first byte have been drawn.

In absolute mode, the first byte contains zero, the second byte contains the number of color indexes that follow, and subsequent bytes contain color indexes in their high- and low-order nibbles, one color index

for each pixel. In absolute mode, each run must be aligned on a word boundary. The end-of-line, end-of-bitmap, and delta escapes also apply to BI\_RLE4.

The following example shows the hexadecimal values of a 4-bit compressed bitmap:

```
03 04 05 06 00 06 45 56 67 00 04 78 00 02 05 01
04 78 00 00 09 1E 00 01
```

This bitmap would expand as follows (single-digit values represent a color index for a single pixel):

```
0 4 0
0 6 0 6 0
4 5 5 6 6 7
7 8 7 8
move current position 5 right and 1 down
7 8 7 8
end of line
1 E 1 E 1 E 1 E 1
end of RLE bitmap
```

See Also

[BITMAPINFO](#)

### CBT\_CREATEWND (3.1)

```
typedef struct tagCBT_CREATEWND {    /* cbtcw */
    CREATESTRUCT FAR* lpcs;
    HWND    hwndInsertAfter;
} CBT_CREATEWND;
```

The CBT\_CREATEWND structure contains information passed to a WH\_CBT hook function before a window is created.

Member	Description
lpcs	Points to a <u>CREATESTRUCT</u> structure that contains initialization parameters for the window about to be created.
hwndInsertAfter	Identifies a window in the window manager's list that will precede the window being created. If this parameter is NULL, the window being created is the topmost window. If this parameter is 1, the window being created is the bottommost window.

See Also

CBTProc, SetWindowsHook

## CBTACTIVATESTRUCT (3.1)

```
typedef struct tagCBTACTIVATESTRUCT { /* cas */
    BOOL    fMouse;
    HWND    hWndActive;
} CBTACTIVATESTRUCT;
```

The CBTACTIVATESTRUCT structure contains information passed to a WH\_CBT hook function before a window is activated.

Member	Description
fMouse	Specifies whether the window is being activated as a result of a mouse click. This value is nonzero if a mouse click is causing the activation. Otherwise, this value is zero.
hWndActive	Identifies the currently active window.

See Also

SetWindowsHook

## CHOOSECOLOR (3.1)

```
#include <commdlg.h>

typedef struct tagCHOOSECOLOR {    /* cc */
    DWORD    lStructSize;
    HWND     hwndOwner;
    HWND     hInstance;
    COLORREF rgbResult;
    COLORREF FAR* lpCustColors;
    DWORD    Flags;
    LPARAM   lCustData;
    UINT     (CALLBACK* lpfnHook)(HWND, UINT, WPARAM, LPARAM);
    LPCSTR   lpTemplateName;
} CHOOSECOLOR;
```

The CHOOSECOLOR structure contains information that the system uses to initialize the system-defined Color dialog box. After the user chooses the OK button to close the dialog box, the system returns information about the user's selection in this structure.

Member	Description								
lStructSize	Specifies the length of the structure, in bytes. This member is filled on input.								
hwndOwner	Identifies the window that owns the dialog box. This member can be any valid window handle, or it should be NULL if the dialog box is to have no owner. If the CC_SHOWHELP flag is set, hwndOwner must identify the window that owns the dialog box. The window procedure for this owner window receives a notification message when the user chooses the Help button. (The identifier for the notification message is the value returned by the <u>RegisterWindowMessage</u> function when HELPMMSGSTRING is passed as its argument.) This member is filled on input.								
hInstance	Identifies a data block that contains the dialog box template specified by the lpTemplateName member. This member is used only if the Flags member specifies the CC_ENABLETEMPLATE or CC_ENABLETEMPLATEHANDLE flag; otherwise, this member is ignored. This member is filled on input.								
rgbResult	Specifies the color that is initially selected when the dialog box is displayed, and specifies the user's color selection after the user has chosen the OK button to close dialog box. If the CC_RGBINIT flag is set in the Flags member before the dialog box is displayed and the value of this member is not among the colors available, the system selects the nearest solid color available. If this member is NULL, the first selected color is black. This member is filled on input and output.								
lpCustColors	Points to an array of 16 doubleword values, each of which specifies the intensities of the red, green, and blue (RGB) components of a custom color box in the dialog box. If the user modifies a color, the system updates the array with the new RGB values. This member is filled on input and output.								
Flags	Specifies the dialog box initialization flags. This member may be a combination of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>CC_ENABLEHOOK</td><td>Enables the hook function specified in the lpfnHook member.</td></tr><tr><td>CC_ENABLETEMPLATE</td><td>Causes the system to use the dialog box template identified by the hInstance member and pointed to by the lpTemplateName member.</td></tr><tr><td>CC_ENABLETEMPLATEHANDLE</td><td>Indicates that the hInstance member identifies a data block that contains a pre-loaded dialog box template. If this flag is</td></tr></tbody></table>	Value	Meaning	CC_ENABLEHOOK	Enables the hook function specified in the lpfnHook member.	CC_ENABLETEMPLATE	Causes the system to use the dialog box template identified by the hInstance member and pointed to by the lpTemplateName member.	CC_ENABLETEMPLATEHANDLE	Indicates that the hInstance member identifies a data block that contains a pre-loaded dialog box template. If this flag is
Value	Meaning								
CC_ENABLEHOOK	Enables the hook function specified in the lpfnHook member.								
CC_ENABLETEMPLATE	Causes the system to use the dialog box template identified by the hInstance member and pointed to by the lpTemplateName member.								
CC_ENABLETEMPLATEHANDLE	Indicates that the hInstance member identifies a data block that contains a pre-loaded dialog box template. If this flag is								

CC_FULLOPEN	specified, the system ignores the lpTemplateName member. Causes the entire dialog box to appear when the dialog box is displayed, including the portion that allows the user to create custom colors. Without this flag, the user must select the Define Custom Color button to see that portion of the dialog box.
CC_PREVENTFULLOPEN	Disables the Define Custom Colors button, preventing the user from creating custom colors.
CC_RGBINIT	Causes the dialog box to use the color specified in the rgbResult member as the initial color selection.
CC_SHOWHELP	Causes the dialog box to show the Help button. If this flag is specified, the hWndOwner member must not be NULL.

These flags are used when the structure is initialized.

lCustData	Specifies application-defined data that the system passes to the hook function pointed to by the lpfnHook member. The system passes a pointer to the CHOOSECOLOR structure in the lParam parameter of the <u>WM_INITDIALOG</u> message; this pointer can be used to retrieve the lCustData member.
lpfnHook	Points to a hook function that processes messages intended for the dialog box. To enable the hook function, an application must specify the CC_ENABLEHOOK value in the Flags member; otherwise, the system ignores this structure member. The hook function must return zero to pass a message that it didn't process back to the dialog box procedure in COMMDLG.DLL. The hook function must return a nonzero value to prevent the dialog box procedure in COMMDLG.DLL from processing a message it has already processed. This member is filled on input.
lpTemplateName	Points to a null-terminated string that specifies the name of the resource file for the dialog box template that is to be substituted for the dialog box template in COMMDLG.DLL. An application can use the MAKEINTRESOURCE macro for numbered dialog box resources. This member is used only if the Flags member specifies the CC_ENABLETEMPLATE flag; otherwise, this member is ignored. This member is filled on input.

#### Comments

Some members of this structure are filled only when the dialog box is created, and some have an initialization value that changes when the user closes the dialog box. Whenever a description in the Members section does not specify how the value of a member is assigned, the value is assigned only when the dialog box is created.

#### See Also

ChooseColor, MAKEINTRESOURCE, RGB

## CHOOSEFONT (3.1)

```
#include <commdlg.h>

typedef struct tagCHOOSEFONT { /* cf */
    DWORD lStructSize;
    HWND hwndOwner;
    HDC hDC;
    LOGFONT FAR* lpLogFont;
    int iPointSize;
    DWORD Flags;
    COLORREF rgbColors;
    LPARAM lCustData;
    UINT (CALLBACK* lpfnHook)(HWND, UINT, WPARAM, LPARAM);
    LPCSTR lpTemplateName;
    HINSTANCE hInstance;
    LPSTR lpszStyle;
    UINT nFontType;
    int nSizeMin;
    int nSizeMax;
} CHOOSEFONT;
```

The CHOOSEFONT structure contains information that the system uses to initialize the system-defined Font dialog box. After the user chooses the OK button to close the dialog box, the system returns information about the user's selection in this structure.

Member	Description						
lStructSize	Specifies the length of the structure, in bytes. This member is filled on input.						
hwndOwner	Identifies the window that owns the dialog box. This member can be any valid window handle, or it should be NULL if the dialog box is to have no owner. If the CF_SHOWHELP flag is set, hwndOwner must identify the window that owns the dialog box. The window procedure for this owner window receives a notification message when the user chooses the Help button. (The identifier for the notification message is the value returned by the <u>RegisterWindowMessage</u> function when HELPPMSGSTRING is passed as its argument.) This member is filled on input.						
hDC	Identifies either the device context or the information context of the printer for which fonts are to be listed in the dialog box. This member is used only if the Flags member specifies the CF_PRINTERFONTS flag; otherwise, this member is ignored. This member is filled on input.						
lpLogFont	Points to a LOGFONT structure. If an application initializes the members of this structure before calling <u>ChooseFont</u> and sets the CF_INITTOLOGFONTSTRUCT flag, the <u>ChooseFont</u> function initializes the dialog box with the font that is the closest possible match. After the user chooses the OK button to close the dialog box, the <u>ChooseFont</u> function sets the members of the LOGFONT structure based on the user's final selection. This member is filled on input and output.						
iPointSize	Specifies the size of the selected font, in tenths of a point. The <u>ChooseFont</u> function sets this value after the user chooses the OK button to close the dialog box.						
Flags	Specifies the dialog box initialization flags. This member can be a combination of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>CF_APPLY</td><td>Specifies that the <u>ChooseFont</u> function should enable the <u>Apply</u> button.</td></tr><tr><td>CF_ANSIONLY</td><td>Specifies that the <u>ChooseFont</u> function should limit font selection to those fonts that use</td></tr></tbody></table>	Value	Meaning	CF_APPLY	Specifies that the <u>ChooseFont</u> function should enable the <u>Apply</u> button.	CF_ANSIONLY	Specifies that the <u>ChooseFont</u> function should limit font selection to those fonts that use
Value	Meaning						
CF_APPLY	Specifies that the <u>ChooseFont</u> function should enable the <u>Apply</u> button.						
CF_ANSIONLY	Specifies that the <u>ChooseFont</u> function should limit font selection to those fonts that use						

	the Windows character set. (If this flag is set, the user cannot select a font that contains only symbols.)
CF_BOTH	Causes the dialog box to list the available printer and screen fonts. The <code>hDC</code> member identifies either the device context or the information context associated with the printer.
CF_TTONLY	Specifies that the <code>ChooseFont</code> function should enumerate and allow the selection of only TrueType fonts.
CF_EFFECTS	Specifies that the <code>ChooseFont</code> function should enable <code>strikeout</code> , <code>underline</code> , and <code>color</code> effects. If this flag is set, the <code>lfStrikeOut</code> and <code>lfUnderline</code> members of the <code>LOGFONT</code> structure and the <code>rgbColors</code> member of the <code>CHOOSEFONT</code> structure can be set before calling <code>ChooseFont</code> . And, if this flag is not set, the <code>ChooseFont</code> function can set these members after the user chooses the OK button to close the dialog box.
CF_ENABLEHOOK	Enables the hook function specified in the <code>lpfnHook</code> member of this structure.
CF_ENABLETEMPLATE	Indicates that the <code>hInstance</code> member identifies a data block that contains the dialog box template pointed to by <code>lpTemplateName</code> .
CF_ENABLETEMPLATEHANDLE	Indicates that the <code>hInstance</code> member identifies a data block that contains a pre-loaded dialog box template. If this flag is specified, the system ignores the <code>lpTemplateName</code> member.
CF_FIXEDPITCHONLY	Specifies that the <code>ChooseFont</code> function should select only monospace fonts.
CF_FORCEFONTEXIST	Specifies that the <code>ChooseFont</code> function should indicate an error condition if the user attempts to select a font or font style that does not exist.
CF_INITTOLOGFONTSTRUCT	Specifies that the <code>ChooseFont</code> function should use the <code>LOGFONT</code> structure pointed to by <code>lpLogFont</code> to initialize the dialog box controls.
CF_LIMITSIZE	Specifies that the <code>ChooseFont</code> function should select only font sizes within the range specified by the <code>nSizeMin</code> and <code>nSizeMax</code> members.



CF_NOFACESEL	Specifies that there is no selection in the Font (face name) combo box. Applications use this flag to support multiple font selections. This flag is set on input and output.
CF_NOOEMFONTS	Specifies that the <u>ChooseFont</u> function should not allow vector-font selections. This flag has the same value as CF_NOVECTORFONTS.
CF_NOSIMULATIONS	Specifies that the <u>ChooseFont</u> function should not allow graphics-device-interface (GDI) font simulations.
CF_NOSIZESEL	Specifies that there is no selection in the Size combo box. Applications use this flag to support multiple size selections. This flag is set on input and output.
CF_NOSTYLESEL	Specifies that there is no selection in the Font Style combo box. Applications use this flag to support multiple style selections. This flag is set on input and output.
CF_NOVECTORFONTS	Specifies that the <u>ChooseFont</u> function should not allow vector-font selections. This flag has the same value as CF_NOOEMFONTS.
CF_PRINTERFONTS	Causes the dialog box to list only the fonts supported by the printer associated with the device context or information context that is identified by the hDC member.
CF_SCALABLEONLY	Specifies that the <u>ChooseFont</u> function should allow the selection of only scalable fonts. (Scalable fonts include vector fonts, some printer fonts, TrueType fonts, and fonts that are scaled by other algorithms or technologies.)
CF_SCREENFONTS	Causes the dialog box to list only the screen fonts supported by the system.
CF_SHOWHELP	Causes the dialog box to show the Help button. If this option is specified, the hwndOwner must not be NULL.
CF_USESTYLE	Specifies that the lpszStyle member points to a buffer that contains a style-description string that the <u>ChooseFont</u> function should use to initialize the Font Style box. When the user chooses the OK button to close the dialog box, the <u>ChooseFont</u> function copies the

	style description for the user's selection to this buffer.						
CF_WYSIWYG	Specifies that the <u>ChooseFont</u> function should allow the selection of only fonts that are available on both the printer and the screen. If this flag is set, the CF_BOTH and CF_SCALABLEONLY flags should also be set.						
rgbColors	These flags may be set when the structure is initialized, except where specified. If the CF_EFFECTS flag is set, this member contains the red, green, and blue (RGB) values the <u>ChooseFont</u> function should use to set the text color. After the user chooses the OK button to close the dialog box, this member contains the RGB values of the color the user selected.						
lCustData	This member is filled on input and output. Specifies application-defined data that the application passes to the hook function. The system passes a pointer to the <u>CHOOSEFONT</u> data structure in the lParam parameter of the <u>WM_INITDIALOG</u> message; the lCustData member can be retrieved using this pointer.						
lpfnHook	Points to a hook function that processes messages intended for the dialog box. To enable the hook function, an application must specify the CF_ENABLEHOOK value in the Flags member; otherwise, the system ignores this structure member. The hook function must return zero to pass a message that it didn't process back to the dialog box procedure in COMMDLG.DLL. The hook function must return a nonzero value to prevent the dialog box procedure in COMMDLG.DLL from processing a message it has already processed.						
lpTemplateName	This member is filled on input. Points to a null-terminated string that specifies the name of the resource file for the dialog box template to be substituted for the dialog box template in COMMDLG.DLL. An application can use the <u>MAKEINTRESOURCE</u> macro for numbered dialog box resources. This member is used only if the Flags member specifies the CF_ENABLETEMPLATE flag; otherwise, this member is ignored.						
hInstance	This member is filled on input. Identifies a data block that contains the dialog box template specified by the lpTemplateName member. This member is used only if the Flags member specifies the CF_ENABLETEMPLATE or the CF_ENABLETEMPLATEHANDLE flag; otherwise, this member is ignored.						
lpszStyle	This member is filled on input. Points to a buffer that contains a style-description string for the font. If the CF_USESTYLE flag is set, the <u>ChooseFont</u> function uses the data in this buffer to initialize the Font Style box. When the user chooses the OK button to close the dialog box, the <u>ChooseFont</u> function copies the string in the Font Style box into this buffer. The buffer pointed to by lpszStyle must be at least LF_FACESIZE bytes long.						
nFontType	This member is filled on input and output. Specifies the type of the selected font. This member can be one or more of the values in the following list:						
	<table border="0"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>BOLD_FONTTYPE</td> <td>Specifies that the font is bold. This value applies only to TrueType fonts. This value corresponds to the value of the ntmFlags member of the <u>NEWTEXTMETRIC</u> structure.</td> </tr> <tr> <td>ITALIC_FONTTYPE</td> <td>Specifies that the font is italic. This value applies only to TrueType fonts. This value corresponds to the value of the ntmFlags</td> </tr> </tbody> </table>	Value	Meaning	BOLD_FONTTYPE	Specifies that the font is bold. This value applies only to TrueType fonts. This value corresponds to the value of the ntmFlags member of the <u>NEWTEXTMETRIC</u> structure.	ITALIC_FONTTYPE	Specifies that the font is italic. This value applies only to TrueType fonts. This value corresponds to the value of the ntmFlags
Value	Meaning						
BOLD_FONTTYPE	Specifies that the font is bold. This value applies only to TrueType fonts. This value corresponds to the value of the ntmFlags member of the <u>NEWTEXTMETRIC</u> structure.						
ITALIC_FONTTYPE	Specifies that the font is italic. This value applies only to TrueType fonts. This value corresponds to the value of the ntmFlags						

		member of the <u>NEWTEXTMETRIC</u> structure.
	PRINTER_FONTTYPE	Specifies that the font is a printer font.
	REGULAR_FONTTYPE	Specifies that the font is neither bold nor italic. This value applies only to TrueType fonts. This value corresponds to the value of the ntmFlags member of the <u>NEWTEXTMETRIC</u> structure.
	SCREEN_FONTTYPE	Specifies that the font is a screen font.
	SIMULATED_FONTTYPE	Specifies that the font is simulated by GDI. This is not set if the CF_NOSIMULATIONS flag is set.
nSizeMin		Specifies the minimum point size that a user can select. The <u>ChooseFont</u> function will recognize this member only if the CF_LIMITSIZE flag is set. This member is filled on input.
nSizeMax		Specifies the maximum point size that a user can select. The <u>ChooseFont</u> function will recognize this member only if the CF_LIMITSIZE flag is set. This member is filled on input.
See Also		
	<u>ChooseFont</u> , <u>LOGFONT</u> , <u>MAKEINTRESOURCE</u>	

## CLASSENTRY (3.1)

```
#include <toolhelp.h>

typedef struct tagCLASSENTRY { /* ce */
    DWORD    dwSize;
    HMODULE  hInst;
    char     szClassName[MAX_CLASSNAME + 1];
    WORD     wNext;
} CLASSENTRY;
```

The CLASSENTRY structure contains the name of a Windows class and a near pointer to the next class in the list. For more information about Windows classes, see the [GetClassInfo](#) function.

Member	Description
dwSize	Specifies the size of the CLASSENTRY structure, in bytes.
hInst	Identifies the instance handle of the task that owns the class. An application needs this handle to call <a href="#">GetClassInfo</a> . The hInst member is really a handle to a module, since Windows classes are owned by modules. Therefore, this hInst will not match the hInst passed as a parameter to the <a href="#">WinMain</a> function of the owning task.
szClassName	Specifies the null-terminated string that contains the class name. An application needs this name to call <a href="#">GetClassInfo</a> .
wNext	Specifies the next class in the list. This member is reserved for internal use by Windows.

See Also

[ClassFirst](#), [ClassNext](#), [GetClassInfo](#)

## CLIENTCREATESTRUCT (3.0)

```
typedef struct tagCLIENTCREATESTRUCT { /* ccs */
    HANDLE hWindowMenu;
    UINT idFirstChild;
} CLIENTCREATESTRUCT;
```

The CLIENTCREATESTRUCT structure contains information about the menu and first multiple document interface (MDI) child window of an MDI client window. An application passes a long pointer to this structure as the lpParam parameter of the CreateWindow function when creating an MDI client window.

Member	Description
hWindowMenu	Identifies the menu handle of the application's Window menu. An application can retrieve this handle from the menu of the MDI frame window by using the <u>GetSubMenu</u> function.
idFirstChild	Specifies the child window identifier of the first MDI child window created. Windows increments the identifier for each additional MDI child window that the application creates, and reassigns identifiers when the application destroys a window to keep the range of identifiers continuous. These identifiers are used in <u>WM_COMMAND</u> messages to the application's MDI frame window when a child window is selected from the Window menu; they should not conflict with any other command identifiers.

See Also  
CreateWindow, GetSubMenu

## COMPAREITEMSTRUCT (3.0)

```
typedef struct tagCOMPAREITEMSTRUCT {    /* cis */
    UINT  CtlType;
    UINT  CtlID;
    HWND  hwndItem;
    UINT  itemID1;
    DWORD itemData1;
    UINT  itemID2;
    DWORD itemData2;
} COMPAREITEMSTRUCT;
```

The COMPAREITEMSTRUCT structure supplies the identifiers and application-supplied data for two items in a sorted owner-drawn combo box or list box.

Whenever an application adds a new item to an owner-drawn combo or list box created with the CBS\_SORT or LBS\_SORT style, Windows sends the owner a WM\_COMPAREITEM message. The lParam parameter of the message contains a long pointer to a COMPAREITEMSTRUCT structure. When the owner receives the message, it compares the two items and returns a value indicating which item sorts before the other.

Member	Description
CtlType	Specifies ODT_LISTBOX (which identifies an owner-drawn list box) or ODT_COMBOBOX (which identifies an owner-drawn combo box).
CtlID	Specifies the identifier of the list box or combo box.
hwndItem	Identifies the control.
itemID1	Specifies the index of the first item in the list box or combo box being compared.
itemData1	Specifies application-supplied data for the first item being compared. (This value was passed as the lParam parameter of the message that added the item to the combo box or list box.)
itemID2	Specifies the index of the second item in the list box or combo box being compared.
itemData2	Specifies application-supplied data for the second item being compared. This value was passed as the lParam parameter of the message that added the item to the combo box or list box.

See Also

WM\_COMPAREITEM

## COMSTAT (3.1)

```
typedef struct tagCOMSTAT { /* cmst */
    BYTE status;
    UINT cbInQue;
    UINT cbOutQue;
} COMSTAT;
```

The COMSTAT structure contains information about a communications device.

Member	Description																
status	Specifies the status of the transmission. This member can be one or more of the following flags:																
	<table><thead><tr><th>Flag</th><th>Meaning</th></tr></thead><tbody><tr><td>CSTF_CTSHOLD</td><td>Specifies whether transmission is waiting for the CTS (clear-to-send) signal to be sent.</td></tr><tr><td>CSTF_DSRHOLD</td><td>Specifies whether transmission is waiting for the DSR (data-set-ready) signal to be sent.</td></tr><tr><td>CSTF_RLSDHOLD</td><td>Specifies whether transmission is waiting for the RLSD (receive-line-signal-detect) signal to be sent.</td></tr><tr><td>CSTF_XOFFHOLD</td><td>Specifies whether transmission is waiting as a result of the XOFF character being received.</td></tr><tr><td>CSTF_XOFFSENT</td><td>Specifies whether transmission is waiting as a result of the XOFF character being transmitted. Transmission halts when the XOFF character is transmitted and used by systems that take the next character as XON, regardless of the actual character.</td></tr><tr><td>CSTF_EOF</td><td>Specifies whether the end-of-file (EOF) character has been received.</td></tr><tr><td>CSTF_TXIM</td><td>Specifies whether a character is waiting to be transmitted.</td></tr></tbody></table>	Flag	Meaning	CSTF_CTSHOLD	Specifies whether transmission is waiting for the CTS (clear-to-send) signal to be sent.	CSTF_DSRHOLD	Specifies whether transmission is waiting for the DSR (data-set-ready) signal to be sent.	CSTF_RLSDHOLD	Specifies whether transmission is waiting for the RLSD (receive-line-signal-detect) signal to be sent.	CSTF_XOFFHOLD	Specifies whether transmission is waiting as a result of the XOFF character being received.	CSTF_XOFFSENT	Specifies whether transmission is waiting as a result of the XOFF character being transmitted. Transmission halts when the XOFF character is transmitted and used by systems that take the next character as XON, regardless of the actual character.	CSTF_EOF	Specifies whether the end-of-file (EOF) character has been received.	CSTF_TXIM	Specifies whether a character is waiting to be transmitted.
Flag	Meaning																
CSTF_CTSHOLD	Specifies whether transmission is waiting for the CTS (clear-to-send) signal to be sent.																
CSTF_DSRHOLD	Specifies whether transmission is waiting for the DSR (data-set-ready) signal to be sent.																
CSTF_RLSDHOLD	Specifies whether transmission is waiting for the RLSD (receive-line-signal-detect) signal to be sent.																
CSTF_XOFFHOLD	Specifies whether transmission is waiting as a result of the XOFF character being received.																
CSTF_XOFFSENT	Specifies whether transmission is waiting as a result of the XOFF character being transmitted. Transmission halts when the XOFF character is transmitted and used by systems that take the next character as XON, regardless of the actual character.																
CSTF_EOF	Specifies whether the end-of-file (EOF) character has been received.																
CSTF_TXIM	Specifies whether a character is waiting to be transmitted.																
cbInQue	Specifies the number of characters in the receive queue.																
cbOutQue	Specifies the number of characters in the transmit queue.																

See Also  
[GetCommError](#)

## CONVCONTEXT (3.1)

```
#include <ddeml.h>

typedef struct tagCONVCONTEXT { /* cc */
    UINT    cb;
    UINT    wFlags;
    UINT    wCountryID;
    int     iCodePage;
    DWORD   dwLangID;
    DWORD   dwSecurity;
} CONVCONTEXT;
```

The CONVCONTEXT structure contains information that makes it possible for applications to share data in several different languages.

Member	Description
cb	Specifies the size, in bytes, of the CONVCONTEXT structure.
wFlags	Specifies conversation-context flags. Currently, no flags are defined for this member.
wCountryID	Specifies the country-code identifier for topic-name and item-name strings.
iCodePage	Specifies the code page for topic-name and item-name strings. Unilingual clients should set this member to CP_WINANSI. An application that uses the OEM character set should set this member to the value returned by the <a href="#">GetKBCodePage</a> function.
dwLangID	Specifies the language identifier for topic-name and item-name strings.
dwSecurity	Specifies a private (application-defined) security code.

See Also

[GetKBCodePage](#)



## CONVINFO (3.1)

```
#include <ddeml.h>

typedef struct tagCONVINFO { /* ci */
    DWORD    cb;
    DWORD    hUser;
    HCONV    hConvPartner;
    HSZhszSvcPartner;
    HSZhszServiceReq;
    HSZhszTopic;
    HSZhszItem;
    UINT     wFmt;
    UINT     wType;
    UINT     wStatus;
    UINT     wConvst;
    UINT     wLastError;
    HCONVLIST hConvList;
    CONVCONTEXT ConvCtxt;
} CONVINFO;
```

The CONVINFO structure contains information about a dynamic data exchange (DDE) conversation.

Member	Description																		
cb	Specifies the length of the structure, in bytes.																		
hUser	Identifies application-defined data.																		
hConvPartner	Identifies the partner application in the DDE conversation. If the partner has not registered itself (by using the <u>DdeInitialize</u> function) to make DDE Management Library (DDEML) function calls, this member is set to 0. An application should not pass this member to any DDEML function except <u>DdeQueryConvInfo</u> .																		
hszSvcPartner	Identifies the service name of the partner application.																		
hszServiceReq	Identifies the service name of the server application that was requested for connection.																		
hszTopic	Identifies the name of the requested topic.																		
hszItem	Identifies the name of the requested item. This member is transaction-specific.																		
wFmt	Specifies the format of the data being exchanged. This member is transaction-specific.																		
wType	Specifies the type of the current transaction. This member is transaction-specific and can be one of the following values:																		
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>XTYP_ADVDATA</td><td>Informs a client that advise data from a server has arrived.</td></tr><tr><td>XTYP_ADVREQ</td><td>Requests that a server send updated data to the client during an advise loop. This transaction results when the server calls the <u>DdePostAdvise</u> function.</td></tr><tr><td>XTYP_ADVSTART</td><td>Requests that a server begin an advise loop with a client.</td></tr><tr><td>XTYP_ADVSTOP</td><td>Notifies a server that an advise loop is ending.</td></tr><tr><td>XTYP_CONNECT</td><td>Requests that a server establish a conversation with a client.</td></tr><tr><td>XTYP_CONNECT_CONFIRM</td><td>Notifies a server that a conversation with a client has been established.</td></tr><tr><td>XTYP_DISCONNECT</td><td>Notifies a server that a conversation has terminated.</td></tr><tr><td>XTYP_ERROR</td><td>Notifies a DDEML application that a critical error has occurred. The DDEML may have insufficient resources to continue.</td></tr></tbody></table>	Value	Meaning	XTYP_ADVDATA	Informs a client that advise data from a server has arrived.	XTYP_ADVREQ	Requests that a server send updated data to the client during an advise loop. This transaction results when the server calls the <u>DdePostAdvise</u> function.	XTYP_ADVSTART	Requests that a server begin an advise loop with a client.	XTYP_ADVSTOP	Notifies a server that an advise loop is ending.	XTYP_CONNECT	Requests that a server establish a conversation with a client.	XTYP_CONNECT_CONFIRM	Notifies a server that a conversation with a client has been established.	XTYP_DISCONNECT	Notifies a server that a conversation has terminated.	XTYP_ERROR	Notifies a DDEML application that a critical error has occurred. The DDEML may have insufficient resources to continue.
Value	Meaning																		
XTYP_ADVDATA	Informs a client that advise data from a server has arrived.																		
XTYP_ADVREQ	Requests that a server send updated data to the client during an advise loop. This transaction results when the server calls the <u>DdePostAdvise</u> function.																		
XTYP_ADVSTART	Requests that a server begin an advise loop with a client.																		
XTYP_ADVSTOP	Notifies a server that an advise loop is ending.																		
XTYP_CONNECT	Requests that a server establish a conversation with a client.																		
XTYP_CONNECT_CONFIRM	Notifies a server that a conversation with a client has been established.																		
XTYP_DISCONNECT	Notifies a server that a conversation has terminated.																		
XTYP_ERROR	Notifies a DDEML application that a critical error has occurred. The DDEML may have insufficient resources to continue.																		

	XTYP_EXECUTE	Requests that a server execute a command sent by a client.
	XTYP_MONITOR	Notifies an application registered as APPCMD_MONITOR of DDE data being transmitted.
	XTYP_POKE	Requests that a server accept unsolicited data from a client.
	XTYP_REGISTER	Notifies other DDEML applications that a server has registered a service name.
	XTYP_REQUEST	Requests that a server send data to a client.
	XTYP_UNREGISTER	Notifies other DDEML applications that a server has unregistered a service name.
	XTYP_WILDCONNECT	Requests that a server establish multiple conversations with the same client.
	XTYP_XACT_COMPLETE	Notifies a client that an asynchronous data transaction has completed.
wStatus	Specifies the status of the current conversation. This member can be a combination of the following values:	
	ST_ADVISE	ST_INLIST
	ST_BLOCKED	ST_ISLOCAL
	ST_BLOCKNEXT	ST_ISSELF
	ST_CLIENT	ST_TERMINATED
	ST_CONNECTED	
wConvst	Specifies the conversation state. This member can be one of the following values:	
	Value	Meaning
	XST_ADVACKRCVD	The advise transaction was just completed.
	XST_ADVDATAACKRCVD	The advise data transaction was just completed.
	XST_ADVDATASENT	Advise data has been sent and is awaiting an acknowledge.
	XST_ADVSENT	An advise transaction is awaiting an acknowledge.
	XST_CONNECTED	The conversation has no active transactions.
	XST_DATARCVD	The requested data has just been received.
	XST_EXECACKRCVD	An execute transaction just completed.
	XST_EXECSENT	An execute transaction is awaiting an acknowledge.
	XST_INCOMPLETE	The last transaction failed.
	XST_INIT1	Mid-initiate state 1
	XST_INIT2	Mid-initiate state 2
	XST_NULL	Pre-initiate state
	XST_POKEACKRCVD	A poke transaction was just completed.
	XST_POKESENT	A poke transaction is awaiting an acknowledge.
	XST_REQSENT	A request transaction is awaiting an acknowledge.
	XST_UNADVACKRCVD	An unadvise transaction just completed.
	XST_UNADVSENT	An unadvise transaction is awaiting an acknowledge.
wLastError	Specifies the error value associated with the last transaction.	
hConvList	If the handle of the current conversation is in a conversation list, identifies the conversation list. Otherwise, this member is NULL.	
ConvCtxt	Specifies the conversation context.	
See Also		

CONVCONTEXT, DdeQueryConvInfo

## CPLINFO (3.1)

```
#include <cpl.h>

typedef struct tagCPLINFO { /* cpli */
    int    idIcon;
    int    idName;
    int    idInfo;
    LONG   lData;
} CPLINFO;
```

The CPLINFO structure contains resource information and a user-defined value for an extensible Control Panel application.

Member	Description
idIcon	Specifies an icon resource identifier for the application icon. This icon is displayed in the Control Panel window.
idName	Specifies a string resource identifier for the application name. The name is the short string displayed below the application icon in the Control Panel window. The name is also displayed on the Settings menu of Control Panel.
idInfo	Specifies a string resource identifier for the application description. The description is the descriptive string displayed at the bottom of the Control Panel window when the application icon is selected.
lData	Specifies user-defined data for the application.

## CREATESTRUCT (2.x)

```
typedef struct tagCREATESTRUCT {    /* cs */
    void FAR* lpCreateParams;
    HINSTANCE hInstance;
    HMENU hMenu;
    HWND hwndParent;
    int cy;
    int cx;
    int y;
    int x;
    LONG style;
    LPCSTR lpszName;
    LPCSTR lpszClass;
    DWORD dwExStyle;
} CREATESTRUCT;
```

The CREATESTRUCT structure defines the initialization parameters passed to the window procedure of an application.

Member	Description
lpCreateParams	Points to data to be used for creating the window.
hInstance	Identifies the module-instance handle of the module that owns the new window.
hMenu	Identifies the menu to be used by the new window.
hwndParent	Identifies the window that owns the new window. This member is NULL if the new window is a top-level window.
cy	Specifies the height of the new window.
cx	Specifies the width of the new window.
y	Specifies the y-coordinate of the upper-left corner of the new window. Coordinates are relative to the parent window if the new window is a child window. Otherwise, the coordinates are relative to the screen origin.
x	Specifies the x-coordinate of the upper-left corner of the new window. Coordinates are relative to the parent window if the new window is a child window. Otherwise, the coordinates are relative to the screen origin.
style	Specifies the style for the new window.
lpszName	Points to a null-terminated string that specifies the name of the new window.
lpszClass	Points to a null-terminated string that specifies the class name of the new window.
dwExStyle	Specifies extended style for the new window.

See Also  
[CreateWindow](#)

## CTLINFO (3.1)

```
#include <custcntl.h>

typedef struct tagCTLINFO {
    UINT    wVersion;
    UINT    wCtlTypes;
    char    szClass[CTLCLASS];
    char    szTitle[CTLTITLE];
    char    szReserved[10];
    CTLTYPE Type[CTLTYPES];
} CTLINFO;
```

The CTLINFO structure defines the class name and version number for a custom control. The CTLINFO structure also contains an array of CTLTYPE structures, each of which lists commonly used combinations of control styles (called variants), with a short description and information about the suggested size.

Member	Description
wVersion	Specifies the control version number. Although you can start your numbering scheme from one digit, most implementations use the lower two digits to represent minor releases.
wCtlTypes	Specifies the number of control types supported by this class. This value should always be greater than zero and less than or equal to the CTLTYPES value.
szClass	Specifies a null-terminated string that contains the control class name supported by the dynamic-link library (DLL). This string should be no longer than the CTLCLASS value.
szTitle	Specifies a null-terminated string that contains various copyright or author information relating to the control library. This string should be no longer than the CTLTITLE value.
Type	Specifies an array of CTLTYPE structures containing information that relates to each of the control types supported by the class. There should be no more elements in the array than specified by the CTLTYPES value.

### Comments

An application calls the ClassInfo function to retrieve basic information about the control library. Based on the information returned, the application can create instances of a control by using one of the supported styles. For example, Dialog Editor calls this function to query a library about the different control styles it can display.

The return value of the ClassInfo function identifies a CTLINFO structure if the function is successful. This information becomes the property of the caller, which must explicitly release it by using the GlobalFree function when the structure is no longer needed.

### See Also

[CTLSTYLE](#), [CTLTYPE](#)

## CTLSTYLE (3.1)

```
#include <custcntl.h>

typedef struct tagCTLSTYLE {
    UINT    wX;
    UINT    wY;
    UINT    wCx;
    UINT    wCy;
    UINT    wId;
    DWORD   dwStyle;
    char    szClass[CTLCLASS];
    char    szTitle[CTLTITLE];
} CTLSTYLE;
```

The CTLSTYLE structure specifies the attributes of the selected control, including the current style flags, location, dimensions, and associated text.

Member	Description
wX	Specifies the x-origin, in screen coordinates, of the control relative to the client area of the parent window.
wY	Specifies the y-origin, in screen coordinates, of the control relative to the client area of the parent window.
wCx	Specifies the current control width, in screen coordinates.
wCy	Specifies the current control height, in screen coordinates.
wId	Specifies the current control identifier. In most cases, you should not allow the user to change this value because Dialog Editor automatically coordinates it with a header file.
dwStyle	Specifies the current control style. The high-order word contains the control-specific flags, and the low-order word contains the Windows-specific flags. You may let the user change these flags to any values supported by your control library.
szClass	Specifies a null-terminated string representing the name of the current control class. You should not allow the user to edit this member, because it is provided for informational purposes only. This string should be no longer than the CTLCLASS value.
szTitle	Specifies with a null-terminated string the text associated with the control. This text is usually displayed inside the control or may be used to store other associated information required by the control. This string should be no longer than the CTLTITLE value.

### Comments

An application calls the ClassStyle function to display a dialog box to edit the style of the selected control. When this function is called, it should display a modal dialog box in which the user can edit the CTLSTYLE members. The user interface of this dialog box should be consistent with that of the predefined controls that Dialog Editor supports.

### See Also

[CTLINFO](#), [CTLTYPE](#)

## CTLTYPE (3.1)

```
#include <custcntl.h>

typedef struct tagCTLTYPE {
    UINT    wType;
    UINT    wWidth;
    UINT    wHeight;
    DWORD   dwStyle;
    char    szDescr[CTLDESCR];
} CTLTYPE;
```

The CTLTYPE structure contains information about a control in a particular class. The CTLINFO structure includes an array of CTLTYPE structures.

Member	Description
wType	Reserved; must be zero.
wWidth	Specifies the suggested width of the control when it is created using Dialog Editor. The width is specified in dialog-box coordinates.
wHeight	Specifies the suggested height of the control when it is created using Dialog Editor. The height is specified in dialog-box coordinates.
dwStyle	Specifies the initial style bits used to obtain this control type. This value includes the control-defined flags in the high-order word and the Windows-defined flags in the low-order word.
szDescr	Defines the name to be used by other development tools when referring to this particular variant of the base control class. Dialog Editor does not refer to this information. This string should not be longer than the CTLDESCR value.

See Also

CTLINFO, CTLSTYLE



## DCB (2.x)

```
typedef struct tagDCB /* dcb */
{
    BYTE Id;
    UINT BaudRate;
    BYTE ByteSize;
    BYTE Parity;
    BYTE StopBits;
    UINT RlsTimeout;
    UINT CtsTimeout;
    UINT DsrTimeout;

    UINT fBinary :1;
    UINT fRtsDisable :1;
    UINT fParity :1;
    UINT fOutxCtsFlow :1;
    UINT fOutxDsrFlow :1;
    UINT fDummy :2;
    UINT fDtrDisable :1;

    UINT fOutX:1;
    UINT fInX :1;
    UINT fPeChar :1;
    UINT fNull:1;
    UINT fChEvt :1;
    UINT fDtrflow :1;
    UINT fRtsflow :1;
    UINT fDummy2 :1;

    char XonChar;
    char XoffChar;
    UINT XonLim;
    UINT XoffLim;
    char PeChar;
    char EofChar;
    char EvtChar;
    UINT TxDelay;
} DCB;
```

The DCB structure defines the control setting for a serial communications device.

Member	Description
Id	Specifies the communication device. This value is set by the device driver. If the most significant bit is set, the DCB structure is for a parallel device.
BaudRate	Specifies the baud rate at which the communications device operates. If the value of the high-order byte is equal to 0xFF, the low-order byte specifies a baud-rate index. The index can be one of the following values: CBR_110      CBR_14400 CBR_4400    CBR_19200 CBR_9200    CBR_38400 CBR_8400    CBR_56000 CBR_6000    CBR_128000 CBR_28000   CBR_256000 CBR_9600 If the high-order byte is not equal to 0xFF, this parameter specifies the actual baud rate.
ByteSize	Specifies the number of bits in the characters transmitted and received. This member can be any number from 4 through 8.
Parity	Specifies the parity scheme to be used. This member can be any one of the following values:

	Value	Meaning
	EVENPARITY	Even
	MARKPARITY	Mark
	NOPARITY	No parity
	ODDPARITY	Odd
StopBits	Specifies the number of stop bits to be used. This member can be any one of the following values:	
	Value	Meaning
	ONESTOPBIT	1 stop bit
	ONE5STOPBITS	1.5 stop bits
	TWOSTOPBITS	2 stop bits
RlsTimeout	Specifies the maximum amount of time, in milliseconds, the device should wait for the RLSD (receive-line-signal-detect) signal. RLSD is also known as the carrier-detect (CD) signal.	
CtsTimeout	Specifies the maximum amount of time, in milliseconds, the device should wait for the CTS (clear-to-send) signal.	
DsrTimeout	Specifies the maximum amount of time, in milliseconds, the device should wait for the DSR (data-set-ready) signal.	
fBinary	Specifies binary mode. In nonbinary mode, the EofChar character is recognized on input and remembered as the end of data.	
fRtsDisable	Specifies whether or not the RTS (request-to-send) signal is disabled. If this member is set, RTS is not used and remains low. If this member is clear, RTS is sent when the device is opened and turned off when the device is closed.	
fParity	Specifies whether parity checking is enabled. If this member is set, parity checking is performed and errors are reported.	
fOutxCtsFlow	Specifies that CTS (clear-to-send) signal is to be monitored for output flow control. If this member is set and CTS is turned off, output is suspended until CTS is again sent.	
fOutxDsrFlow	Specifies that the DSR (data-set-ready) signal is to be monitored for output flow control. If this member is set and DSR is turned off, output is suspended until DSR is again sent.	
fDummy	Reserved.	
fDtrDisable	Specifies whether the DTR (data-terminal-ready) signal is disabled. If this member is set, DTR is not used and remains low. If this member is clear, DTR is sent when the device is opened and turned off when the device is closed.	
fOutX	Specifies that XON/XOFF flow control is used during transmission. If this member is set, transmission stops when the XoffChar character is received and starts again when the XonChar character is received.	
fInX	Specifies that XON/XOFF flow control is used during reception. If this member is set, the XonChar character is sent when the reception queue comes within XoffLim characters of being full and the XonChar character is sent when the reception queue comes within XonLim characters of being empty.	
fPeChar	Specifies that characters received with parity errors are to be replaced with the character specified by this member. This member must be set for the replacement to occur.	
fNull	Specifies that received null characters are to be discarded.	
fChEvt	Specifies that reception of the EvtChar character is to be flagged as an event.	
fDtrflow	Specifies that the DTR (data-terminal-ready) signal is to be used for reception flow control. If this member is set, DTR is turned off when the reception queue comes within XoffLim characters of being full and sent when the reception queue comes within XonLim characters of being empty.	
fRtsflow	Specifies that the RTS (ready-to-send) signal is to be used for reception flow control. If this member is set, RTS is turned off when the reception queue comes within XoffLim characters of being full, and sent when the reception queue comes within XonLim characters of being empty.	
fDummy2	Reserved.	
XonChar	Specifies the value of the XON character for both transmission and reception.	

XoffChar	Specifies the value of the XOFF character for both transmission and reception.
XonLim	Specifies the minimum number of characters allowed in the reception queue before the XON character is sent.
XoffLim	Specifies the maximum number of characters allowed in the reception queue before the XOFF character is sent. The value of the XoffLim member is subtracted from the size of the reception queue, in bytes, to calculate the maximum number of characters allowed.
PeChar	Specifies the value of the character used to replace characters received with a parity error.
EofChar	Specifies the value of the character used to signal the end of data.
EvtChar	Specifies the value of the character used to signal an event.
TxDelay	Not currently used.

See Also

[BuildCommDCB](#), [GetCommState](#), [SetCommState](#)

### Windows 3.1 changes

The BaudRate member can specify either the actual baud rate or a baud-rate index. If the high-order byte is equal to 0xFF, the low-order byte specifies one of the following baud-rate index values:

- CBR\_110
- CBR\_300
- CBR\_600
- CBR\_1200
- CBR\_2400
- CBR\_4800
- CBR\_9600
- CBR\_14400
- CBR\_19200
- CBR\_38400
- CBR\_56000
- CBR\_128000
- CBR\_256000

If the high-order byte is not equal to 0xFF, this parameter specifies the actual baud rate.

## DDEACK (2.x)

```
#include <dde.h>

typedef struct tagDDEACK { /* ddeack */
    WORD bAppReturnCode:8,
    reserved:6,
    fBusy:1,
    fAck:1;
} DDEACK;
```

The DDEACK structure contains status flags that a DDE application passes to its partner as part of the WM\_DDE\_ACK message. The flags provide details about the application's response to a WM\_DDE\_ADVISE, WM\_DDE\_DATA, WM\_DDE\_EXECUTE, WM\_DDE\_REQUEST, WM\_DDE\_POKE, or WM\_DDE\_UNADVISE message.

Member	Description
bAppReturnCode	Specifies an application-defined return code.
fBusy	Indicates whether the application was busy and unable to respond to the partner's message at the time the message was received. A nonzero value indicates the server was busy and unable to respond. The fBusy member is defined only when the fAck member is zero.
fAck	Indicates whether the application accepted the message from its partner. A nonzero value indicates the server accepted the message.

See Also

WM\_DDE\_ACK, WM\_DDE\_ADVISE, WM\_DDE\_DATA, WM\_DDE\_EXECUTE, WM\_DDE\_REQUEST, WM\_DDE\_POKE, WM\_DDE\_UNADVISE, DDEADVISE, DDEDATA, DDEPOKE

## DDEADVISE (2.x)

```
#include <dde.h>

typedef struct tagDDEADVISE { /* ddeadv */
    WORD reserved:14,
    fDeferUpd:1,
    fAckReq:1;
    short cfFormat;
} DDEADVISE;
```

The DDEADVISE structure contains flags that specify how a server should send data to a client during an advise loop. A client passes the handle of a DDEADVISE structure to a server as part of a WM\_DDE\_ADVISE message.

Member	Description
fDeferUpd	Indicates whether the server should defer sending updated data to the client. A nonzero value tells the server to send a <u>WM_DDE_DATA</u> message with a NULL data handle whenever the data item changes. In response, the client can post a <u>WM_DDE_REQUEST</u> message to the server to obtain a handle to the updated data.
fAckReq	Indicates whether the server should set the fAckReq flag in the <u>WM_DDE_DATA</u> messages that it posts to the client. A nonzero value tells the server to set the fAckReq bit.
cfFormat	Specifies the client application's preferred data format. The format must be a standard or registered clipboard format. The following standard clipboard formats may be used: CF_BITMAP                   CF_OEMTEXT CF_DCF_OEMTEXT           CF_PALETTE CF_DCF_PALETTE           CF_PENDATA CF_DCF_PENDATA           CF_SYLK CF_DCF_SYLK               CF_TEXT CF_DCF_TEXT               CF_TIFF CF_METAFILEPICT

See Also

WM\_DDE\_ADVISE, WM\_DDE\_DATA, WM\_DDE\_UNADVISE, DDEACK, DDEDATA, DDEPOKE

## DDEDATA (2.x)

```
#include <dde.h>

typedef struct tagDDEDATA { /* ddedat */
    WORD    unused:12,
    fResponse:1,
    fRelease:1,
    reserved:1,
    fAckReq:1;
    short   cfFormat;
    BYTE    Value[1];
} DDEDATA;
```

The DDEDATA structure contains the data and information about the data sent as part of a WM\_DDE\_DATA message.

Member	Description
fResponse	Indicates whether the application receiving the <u>WM_DDE_DATA</u> message should acknowledge receipt of the data by sending a <u>WM_DDE_ACK</u> message. A nonzero value indicates the application should send the acknowledgment.
fRelease	Indicates if the application receiving the <u>WM_DDE_POKE</u> message should free the data. A nonzero value indicates the data should be freed.
fAckReq	Indicates whether the data was sent in response to a <u>WM_DDE_REQUEST</u> message or a <u>WM_DDE_ADVISE</u> message. A nonzero value indicates the data was sent in response to a <u>WM_DDE_REQUEST</u> message.
cfFormat	Specifies the format of the data. The format should be a standard or registered clipboard format. The following standard clipboard formats may be used: CF_BITMAP                    CF_OEMTEXT CF_DCF_OEMTEXT              CF_PALETTE CF_DCF_PALETTE              CF_PENDATA CF_DCF_PENDATA              CF_SYLK CF_DCF_SYLK                  CF_TEXT CF_DCF_TEXT                  CF_TIFF CF_METAFILEPICT

See Also

WM\_DDE\_ACK, WM\_DDE\_ADVISE, WM\_DDE\_DATA, WM\_DDE\_POKE, WM\_DDE\_REQUEST, DDEACK, DDEADVISE, DDEPOKE

## DDEPOKE (2.x)

```
#include <dde.h>

typedef struct tagDDEPOKE { /* ddepok */
    WORD unused:13,
    fRelease:1,
    fReserved:2;
    short cfFormat;
    BYTE Value[1];
} DDEPOKE;
```

The DDEPOKE structure contains the data and information about the data sent as part of a WM\_DDE\_POKE message.

Member	Description
fRelease	Indicates if the application receiving the <u>WM_DDE_POKE</u> message should free the data. A nonzero value specifies the data should be freed.
cfFormat	Specifies the format of the data. The format should be a standard or registered clipboard format. The following standard clipboard formats may be used: CF_BITMAP                    CF_OEMTEXT CF_DCF_OEMTEXT              CF_PALETTE CF_DCF_PALETTE              CF_PENDATA CF_DCF_PENDATA              CF_SYLK CF_DCF_SYLK                 CF_TEXT CF_DCF_TEXT                 CF_TIFF CF_METAFILEPICT
Value	Contains the data. The size of this array depends on the value of the cfFormat member.

See Also

WM\_DDE\_POKE, DDEACK, DDEADVISE, DDEDATA



## DEBUGHOOKINFO (3.1)

```
typedef struct tagDEBUGHOOKINFO { /* dh */
    HMODULE hModuleHook;
    LPARAM reserved;
    LPARAM lParam;
    WPARAM wParam;
    int code;
} DEBUGHOOKINFO;
```

The DEBUGHOOKINFO structure contains debugging information.

Member	Description
hModuleHook	Identifies the module containing the filter function.
reserved	Not used.
lParam	Specifies the value to be passed to the hook in the lParam parameter of the <u>DebugProc</u> callback function.
wParam	Specifies the value to be passed to the hook in the wParam parameter of the <u>DebugProc</u> callback function.
code	Specifies the value to be passed to the hook in the code parameter of the <u>DebugProc</u> callback function.

See Also

DebugProc, SetWindowsHook

## DELETEITEMSTRUCT (3.0)

```
typedef struct tagDELETEITEMSTRUCT {    /* deli */
    UINT  CtlType;
    UINT  CtlID;
    UINT  itemID;
    HWND  hwndItem;
    DWORD itemData;
} DELETEITEMSTRUCT;
```

The DELETEITEMSTRUCT structure describes a deleted owner-drawn list-box or combo-box item. When an item is removed from the list box or combo box or when the list box or combo box is destroyed, Windows sends the WM\_DELETEITEM message to the owner for each deleted item. The lParam parameter of the message contains a pointer to this structure.

Member	Description
CtlType	Contains ODT_LISTBOX (which specifies an owner-drawn list box) or ODT_COMBOBOX (which specifies an owner-drawn combo box).
CtlID	Contains the control identifier for the list box or combo box.
itemID	Contains the index of the item in the list box or combo box being removed.
hwndItem	Contains the window handle of the control.
itemData	Contains the value passed to the control in the lParam parameter of the <u>LB_INSERTSTRING</u> , <u>LB_ADDSTRING</u> , <u>CB_INSERTSTRING</u> , or <u>CB_ADDSTRING</u> message when the item was added to the list box.

See Also

WM\_DELETEITEM

## DEVMODE (3.0)

```
#include <print.h>

typedef struct tagDEVMODE { /* dm */
    char    dmDeviceName[CCHDEVICENAME];
    UINT    dmSpecVersion;
    UINT    dmDriverVersion;
    UINT    dmSize;
    UINT    dmDriverExtra;
    DWORD   dmFields;
    int     dmOrientation;
    int     dmPaperSize;
    int     dmPaperLength;
    int     dmPaperWidth;
    int     dmScale;
    int     dmCopies;
    int     dmDefaultSource;
    int     dmPrintQuality;
    int     dmColor;
    int     dmDuplex;
    int     dmYResolution;
    int     dmTTOption;
} DEVMODE;
```

The DEVMODE structure contains information about a printer driver's initialization and environment data. An application passes this structure to the DeviceCapabilities and ExtDeviceMode functions.

Member	Description
dmDeviceName	Specifies the name of the device the driver supports--for example, "PCL/HP LaserJet" in the case of the Hewlett-Packard LaserJet. Each driver has a unique string.
dmSpecVersion	Specifies the version number of the DEVMODE structure. For Windows version 3.1, this value should be 0x30A.
dmDriverVersion	Specifies the printer driver version number assigned by the printer driver developer.
dmSize	Specifies the size, in bytes, of the DEVMODE structure. (This value does not include the optional dmDriverData member for device-specific data, which can follow the structure.) If an application manipulates only the driver-independent portion of the data, it can use this member to find out the length of the structure without having to account for different versions.
dmDriverExtra	Specifies the size, in bytes, of the optional dmDriverData member for device-specific data, which can follow the structure. If an application does not use device-specific information, it should set this member to zero.
dmFields	Specifies a set of flags that indicate which of the remaining members in the DEVMODE structure have been initialized. It can be any combination (or it can be none) of the following values:

Constant	Value
DM_ORIENTATION	0x0000001L
DM_PAPERSIZE	0x0000002L
DM_PAPERLENGTH	0x0000004L
DM_PAPERWIDTH	0x0000008L
DM_SCALE	0x0000010L
DM_COPIES	0x0000100L
DM_DEFAULTSOURCE	0x0000200L
DM_PRINTQUALITY	0x0000400L
DM_COLOR	0x0000800L
DM_DUPLEX	0x0001000L
DM_YRESOLUTION	0x0002000L
DM_TTOPTION	0x0004000L

A printer driver supports only those members that are appropriate for the printer technology.

dmOrientation

Specifies the orientation of the paper. It can be either DMORIENT\_PORTRAIT or DMORIENT\_LANDSCAPE.

dmPaperSize

Specifies the size of the paper to print on. This member may be set to zero if the length and width of the paper are specified by the dmPaperLength and dmPaperWidth members, respectively. Otherwise, the dmPaperSize member can be set to one of the following predefined values:

Value	Meaning
DMPAPER_FIRST	DMPAPER_LETTER
DMPAPER_LETTER	Letter, 8 1/2 x 11 in.
DMPAPER_LETTERS <small>SMALL</small>	Letter Small, 8 1/2 x 11 in.
DMPAPER_TABLOID	Tabloid, 11 x 17 in.
DMPAPER_LEDGER	Ledger, 17 x 11 in.
DMPAPER_LEGAL	Legal, 8 1/2 x 14 in.
DMPAPER_STATEMENT	Statement, 5 1/2 x 8 1/2 in.
DMPAPER_EXECUTIVE	Executive, 7 1/2 x 10 1/2 in.
DMPAPER_A3	A3, 297 x 420 mm
DMPAPER_A4	A4, 210 x 297 mm
DMPAPER_A4 <small>SMALL</small>	A4 Small, 210 x 297 mm
DMPAPER_A5	A5, 148 x 210 mm
DMPAPER_B4	B4, 250 x 354 mm
DMPAPER_B5	B5, 182 x 257 mm
DMPAPER_FOLIO	Folio, 8 1/2 x 13 in.
DMPAPER_QUARTO	Quarto, 215 x 275 mm
DMPAPER_10X14	10 x 14 in.
DMPAPER_11X17	11 x 17 in.
DMPAPER_NOTE	Note, 8 1/2 x 11 in.
DMPAPER_ENV_9	Envelope #9, 3 7/8 x 8 7/8 in.
DMPAPER_ENV_10	Envelope #10, 4 1/8 x 9 1/2 in.
DMPAPER_ENV_11	Envelope #11, 4 1/2 x 10 3/8 in.
DMPAPER_ENV_12	Envelope #12, 4 1/2 x 11 in.
DMPAPER_ENV_14	Envelope #14, 5 x 11 1/2 in.
DMPAPER_CSHEET	C size sheet
DMPAPER_DSHEET	D size sheet
DMPAPER_ESHEET	E size sheet
DMPAPER_ENV_DL	Envelope DL, 110 x 220 mm
DMPAPER_ENV_C3	Envelope C3, 324 x 458 mm
DMPAPER_ENV_C4	Envelope C4, 229 x 324 mm
DMPAPER_ENV_C5	Envelope C5, 162 x 229 mm
DMPAPER_ENV_C6	Envelope C6, 114 x 162 mm
DMPAPER_ENV_C65	Envelope C65, 114 x 229 mm
DMPAPER_ENV_B4	Envelope B4, 250 x 353 mm
DMPAPER_ENV_B5	Envelope B5, 176 x 250 mm
DMPAPER_ENV_B6	Envelope B6, 176 x 125 mm
DMPAPER_ENV_ITALY	Envelope, 110 x 230 mm
DMPAPER_ENV_MONARCH	Envelope Monarch, 3 7/8 x 7 1/2 in.
DMPAPER_ENV_PERSONAL	Envelope, 3 5/8 x 6 1/2 in.

	DMPAPER_FANFOLD_US	U.S. Standard Fanfold, 14 7/8 x 11 in.
	DMPAPER_FANFOLD_STD_GERMAN	German Standard Fanfold, 8 1/2 x 12 in.
	DMPAPER_FANFOLD_LGL_GERMAN	German Legal Fanfold, 8 1/2 x 13 in.
	DMPAPER_LAST	German Legal Fanfold, 8 1/2 x 13 in.
	DMPAPER_USER	User-defined
dmPaperLength	Specifies a paper length, in tenths of a millimeter. This parameter overrides the paper length specified by the dmPaperSize member, either for custom paper sizes or for such devices as dot-matrix printers that can print on a variety of page sizes.	
dmPaperWidth	Specifies a paper width, in tenths of a millimeter. This parameter overrides the paper width specified by the dmPaperSize member.	
dmScale	Specifies the factor by which the printed output is to be scaled. The apparent page size is scaled from the physical page size by a factor of dmScale/100. For example, a letter-size paper with a dmScale value of 50 would contain as much data as a page of size 17 by 22 inches because the output text and graphics would be half their original height and width.	
dmCopies	Specifies the number of copies printed if the device supports multiple-page copies.	
dmDefaultSource	Specifies the default bin from which the paper is fed. The application can override this value by using the GETSETPAPERBINS escape. This member can be one of the following values:	
	DMBIN_AUTO	DMBIN_LOWER
	DMBIN_CASSETTE	DMBIN_MANUAL
	DMBIN_ENVELOPE	DMBIN_MIDDLE
	DMBIN_ENVMANUAL	DMBIN_ONLYONE
	DMBIN_FIRST	DMBIN_SMALLFMT
	DMBIN_LARGECAPACITY	DMBIN_TRACTOR
	DMBIN_LARGEFORMAT	DMBIN_UPPER
	DMBIN_LAST	
	A range of values is reserved for device-specific bins. To be consistent with initialization information, the GETSETPAPERBINS and ENUMPAPERBINS escapes use these values.	
dmPrintQuality	Specifies the printer resolution. Following are the four predefined device-independent values:	
	DMRES_HIGH (-4)	
	DMRES_MEDIUM (-3)	
	DMRES_LOW (-2)	
	DMRES_DRAFT (-1)	
	If a positive value is given, it specifies the number of dots per inch (DPI) and is therefore device-dependent.	
	If the printer initializes the dmYResolution member, the dmPrintQuality member specifies the x-resolution of the printer, in dots per inch.	
dmColor	Specifies whether a color printer is to render color or monochrome output. Possible values are:	
	DMCOLOR_COLOR (1)	
	DMCOLOR_MONOCHROME (2)	
dmDuplex	Specifies duplex (double-sided) printing for printers capable of duplex printing. This member can be one of the following values:	
	DMDUP_SIMPLEX (1)	
	DMDUP_HORIZONTAL (2)	
	DMDUP_VERTICAL (3)	

dmYResolution Specifies the y-resolution of the printer, in dots per inch. If the printer initializes this member, the dmPrintQuality member specifies the x-resolution of the printer, in dots per inch.

dmTTOption Specifies how TrueType fonts should be printed. It can be one of the following values:

Value	Meaning
DMTT_BITMAP	Print TrueType fonts as graphics. This is the default action for dot-matrix printers.
DMTT_DOWNLOAD	Download TrueType fonts as soft fonts. This is the default action for Hewlett-Packard printers that use Printer Control Language (PCL).
DMTT_SUBDEV	Substitute device fonts for TrueType fonts. This is the default action for PostScript printers.

#### Comments

Only drivers that are fully updated for Windows versions 3.0 and later and that export the ExtDeviceMode function use the DEVMODE structure.

An application can retrieve the paper sizes and names supported by a printer by calling the DeviceCapabilities function with the DC\_PAPERS, DC\_PAPERSIZE, and DC\_PAPERNAMEs values.

Before setting the value of the dmTTOption member, applications should find out how a printer driver can use TrueType fonts by calling the DeviceCapabilities function with the DC\_TRUETYPE value.

Drivers can add device-specific data immediately following the DEVMODE structure.

See Also

DeviceCapabilities, ExtDeviceMode

## DEVNAMES (3.1)

```
#include <commdlg.h>

typedef struct tagDEVNAMES {      /* dn */
    UINT wDriverOffset;
    UINT wDeviceOffset;
    UINT wOutputOffset;
    UINT wDefault;
    /* optional data may appear here */
} DEVNAMES;
```

The DEVNAMES structure contains offsets to strings that specify the driver, name, and output port of a printer. The PrintDlg function uses these strings to initialize controls in the system-defined Print dialog box. When the user chooses the OK button to close the dialog box, information about the selected printer is returned in this structure.

Member	Description
wDriverOffset	Specifies the offset from the beginning of the structure to a null-terminated string that specifies the Microsoft®MS-DOS®filename (without extension) of the device driver. On input, this string is used to set which printer to initially display in the dialog box.
wDeviceOffset	Specifies the offset from the beginning of the structure to the null-terminated string that specifies the name of the device. This string cannot exceed 32 bytes in length, including the null character, and must be identical to the dmDeviceName member of the <u>DEVMODE</u> structure.
wOutputOffset	Specifies the offset from the beginning of the structure to the null-terminated string that specifies the MS-DOS device name for the physical output medium (output port).
wDefault	Specifies whether the strings specified in the DEVNAMES structure identify the default printer. It is used to verify that the default printer has not changed since the last print operation. On input, this member can be set to DN_DEFAULTPRN. If the DN_DEFAULTPRN flag is set, the other values in the DEVNAMES structure are checked against the current default printer.  On output, the wDefault member is changed only if the Print Setup dialog box was displayed and the user chose the OK button to close it. If the default printer was selected, the DN_DEFAULTPRN flag is set. If a printer is specifically selected, the flag is not set. All other bits in this member are reserved for internal use by the dialog box procedure of the Print dialog box.

See Also  
PrintDlg, DEVMODE

### DOCINFO (3.1)

```
typedef struct { /* di */
    int cbSize;
    LPCSTR lpszDocName;
    LPCSTR lpszOutput;
} DOCINFO;
```

The DOCINFO structure contains the input and output filenames used by the StartDoc function.

Member	Description
cbSize	Specifies the size of the structure, in bytes.
lpszDocName	Points to a null-terminated string specifying the name of the document. This string must not be longer than 32 characters, including the null terminating character.
lpszOutput	Points to a null-terminated string specifying the name of an output file. This allows a print job to be redirected to a file. If this value is NULL, output goes to the device for the specified device context.

See Also  
StartDoc



## DRAWITEMSTRUCT (3.0)

```
typedef struct tagDRAWITEMSTRUCT { /* ditm */
    UINT CtlType;
    UINT CtlID;
    UINT itemID;
    UINT itemAction;
    UINT itemState;
    HWND hwndItem;
    HDC hDC;
    RECT rcItem;
    DWORD itemData;
} DRAWITEMSTRUCT;
```

The DRAWITEMSTRUCT structure provides information the owner needs to determine how to paint an owner-drawn control. The owner of the owner-drawn control receives a pointer to this structure as the lParam parameter of the WM\_DRAWITEM message.

Member	Description												
CtlType	Specifies the control type. The values for control types follow: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>ODT_BUTTON</td><td>Owner-drawn button</td></tr><tr><td>ODT_COMBOBOX</td><td>Owner-drawn combo box</td></tr><tr><td>ODT_LISTBOX</td><td>Owner-drawn list box</td></tr><tr><td>ODT_MENU</td><td>Owner-drawn menu</td></tr></tbody></table>	Value	Meaning	ODT_BUTTON	Owner-drawn button	ODT_COMBOBOX	Owner-drawn combo box	ODT_LISTBOX	Owner-drawn list box	ODT_MENU	Owner-drawn menu		
Value	Meaning												
ODT_BUTTON	Owner-drawn button												
ODT_COMBOBOX	Owner-drawn combo box												
ODT_LISTBOX	Owner-drawn list box												
ODT_MENU	Owner-drawn menu												
CtlID	Specifies the control identifier for a combo box, list box or button. This member is not used for a menu.												
itemID	Specifies the menu-item identifier for a menu or the index of the item in a list box or combo box. For an empty list box or combo box, this member is a negative value. This allows the application to draw only the focus rectangle at the coordinates specified by the rcItem member even though there are no items in the control. This indicates to the user whether the list box or combo box has input focus. The setting of the bits in the itemAction member determines whether the rectangle is to be drawn as though the list box or combo box has input focus.												
itemAction	Specifies the drawing action required. This member is one or more of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>ODA_DRAWENTIRE</td><td>Bit is set when the entire control needs to be drawn.</td></tr><tr><td>ODA_FOCUS</td><td>Bit is set when the control gains or loses input focus. The itemState member should be checked to determine whether the control has focus.</td></tr><tr><td>ODA_SELECT</td><td>Bit is set when only the selection status has changed. The itemState member should be checked to determine the new selection state.</td></tr></tbody></table>	Value	Meaning	ODA_DRAWENTIRE	Bit is set when the entire control needs to be drawn.	ODA_FOCUS	Bit is set when the control gains or loses input focus. The itemState member should be checked to determine whether the control has focus.	ODA_SELECT	Bit is set when only the selection status has changed. The itemState member should be checked to determine the new selection state.				
Value	Meaning												
ODA_DRAWENTIRE	Bit is set when the entire control needs to be drawn.												
ODA_FOCUS	Bit is set when the control gains or loses input focus. The itemState member should be checked to determine whether the control has focus.												
ODA_SELECT	Bit is set when only the selection status has changed. The itemState member should be checked to determine the new selection state.												
itemState	Specifies the visual state of the item after the current drawing action takes place; that is, if a menu item is to be grayed, the state flag ODS_GRAYED will be set. Following are the state flags: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>ODS_CHECKED</td><td>Bit is set if the menu item is to be checked. This bit is used only in a menu.</td></tr><tr><td>ODS_DISABLED</td><td>Bit is set if the item is to be drawn as disabled.</td></tr><tr><td>ODS_FOCUS</td><td>Bit is set if the item has input focus.</td></tr><tr><td>ODS_GRAYED</td><td>Bit is set if the item is to be grayed. This bit is used only in a menu.</td></tr><tr><td>ODS_SELECTED</td><td>Bit is set if the item's status is selected.</td></tr></tbody></table>	Value	Meaning	ODS_CHECKED	Bit is set if the menu item is to be checked. This bit is used only in a menu.	ODS_DISABLED	Bit is set if the item is to be drawn as disabled.	ODS_FOCUS	Bit is set if the item has input focus.	ODS_GRAYED	Bit is set if the item is to be grayed. This bit is used only in a menu.	ODS_SELECTED	Bit is set if the item's status is selected.
Value	Meaning												
ODS_CHECKED	Bit is set if the menu item is to be checked. This bit is used only in a menu.												
ODS_DISABLED	Bit is set if the item is to be drawn as disabled.												
ODS_FOCUS	Bit is set if the item has input focus.												
ODS_GRAYED	Bit is set if the item is to be grayed. This bit is used only in a menu.												
ODS_SELECTED	Bit is set if the item's status is selected.												
hwndItem	Specifies the window handle of the control for combo boxes, list boxes, and buttons. For menus, it contains the handle of the menu (HMENU) containing the item.												

**hDC** Identifies a device context; this device context must be used when performing drawing operations on the control.

**rcItem** Specifies a rectangle in the device context identified by the **hDC** member that defines the boundaries of the control to be drawn. Windows automatically clips anything the owner draws in the device context for combo boxes, list boxes, and buttons, but it does not clip menu items. When drawing menu items, it must ensure that the owner does not draw outside the boundaries of the rectangle defined by the **rcItem** member.

**itemData** Contains the value last assigned to the list box or combo box by an **LB\_SETITEMDATA** or **CB\_SETITEMDATA** message. If the list box or combo box has the **LBS\_HASSTRINGS** or **CBS\_HASSTRINGS** style, this value is initially zero. Otherwise, this value is initially the value that was passed to the list box or combo box in the **lParam** parameter of one of the following messages:

**CB\_ADDSTRING**  
**CB\_INSERTSTRING**  
**LB\_ADDSTRING**  
**LB\_INSERTSTRING**

## DRIVERINFOSTRUCT (3.1)

```
typedef struct tagDRIVERINFOSTRUCT { /* drvinfo */
    UINT length;
    HDRVRhDriver;
    HINSTANCE hModule;
    char szAliasName[128];
} DRIVERINFOSTRUCT;
```

The DRIVERINFOSTRUCT structure contains basic information about an installable device driver.

Member	Description
length	Specifies the size of the DRIVERINFOSTRUCT structure.
hDriver	Identifies an instance of the installable driver.
hModule	Identifies an installable driver module.
szAliasName	Points to a null-terminated string that specifies the driver name or an alias under which the driver was loaded.

See Also

[GetDriverInfo](#)

### DRVCONFIGINFO (3.1)

```
typedef struct tagDRVCONFIGINFO {    /* drvci */
    DWORD    dwDCISize;
    LPCSTR   lpszDCISectionName;
    LPCSTR   lpszDCIAliasName;
} DRVCONFIGINFO;
```

The DRVCONFIGINFO structure contains information about the entries for an installable device driver in the SYSTEM.INI file. This structure is sent in the lParam parameter of the DRV\_CONFIGURE and DRV\_INSTALL installable-driver messages.

Member	Description
dwDCISize	Specifies the size of the DRVCONFIGINFO structure.
lpszDCISectionName	Points to a null-terminated string that specifies the name of the section in the SYSTEM.INI file where driver information is recorded.
lpszDCIAliasName	Points to a null-terminated string that specifies the driver name or an alias under which the driver was loaded.

See Also

DRV\_CONFIGURE, DRV\_INSTALL

## EVENTMSG (2.x)

```
typedef struct tagEVENTMSG {      /* em */
    UINT  message;
    UINT  paramL;
    UINT  paramH;
    DWORD time;
} EVENTMSG;
```

The EVENTMSG structure contains information from the Windows application queue. This structure is used to store message information for the JournalPlaybackProc callback function.

Member	Description
message	Specifies the message.
paramL	Specifies additional information about the message. The exact meaning depends on the message value.
paramH	Specifies additional information about the message. The exact meaning depends on the message value.
time	Specifies the time at which the message was posted.

See Also

JournalPlaybackProc, SetWindowsHook, MSG

## FINDREPLACE (3.1)

```
#include <commdlg.h>

typedef struct tagFINDREPLACE {    /* fr */
    DWORD lStructSize;
    HWND hwndOwner;
    HINSTANCE hInstance;
    DWORD Flags;
    LPSTR lpstrFindWhat;
    LPSTR lpstrReplaceWith;
    UINT wFindWhatLen;
    UINT wReplaceWithLen;
    LPARAM lCustData;
    UINT (CALLBACK* lpfnHook)(HWND, UINT, WPARAM, LPARAM);
    LPCSTR lpTemplateName;
} FINDREPLACE;
```

The FINDREPLACE structure contains information that the system uses to initialize a system-defined Find dialog box or Replace dialog box. After the user chooses the OK button to close the dialog box, the system returns information about the user's selections in this structure.

Member	Description										
lStructSize	Specifies the length of the structure, in bytes. This member is filled on input.										
hwndOwner	Identifies the window that owns the dialog box. This member can be any valid window handle, but it must not be NULL.  If the FR_SHOWHELP flag is set, hwndOwner must identify the window that owns the dialog box. The window procedure for this owner window receives a notification message when the user chooses the Help button. (The identifier for the notification message is the value returned by the <u>RegisterWindowMessage</u> function when HELPMMSGSTRING is passed as its argument.)										
hInstance	This member is filled on input. Identifies a data block that contains a dialog box template specified by the lpTemplateName member. This member is only used if the Flags member specifies the FR_ENABLETEMPLATE or the FR_ENABLETEMPLATEHANDLE flag; otherwise, this member is ignored.										
Flags	This member is filled on input. Specifies the dialog box initialization flags. This member can be a combination of the following values:										
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>FR_DIALOGTERM</td><td>Indicates the dialog box is closing. The window handle returned by the <u>FindText</u> or <u>ReplaceText</u> function is no longer valid after this bit is set. This flag is set by the system.</td></tr><tr><td>FR_DOWN</td><td>Sets the direction of searches through a document. If the flag is set, the search direction is down; if the flag is clear, the search direction is up. Initially, this flag specifies the state of the Up and Down buttons; after the user chooses the OK button to close the dialog box, this flag specifies the user's selection.</td></tr><tr><td>FR_ENABLEHOOK</td><td>Enables the hook function specified in the lpfnHook member of this structure. This flag can be set on input.</td></tr><tr><td>FR_ENABLETEMPLATE</td><td>Causes the system to use the dialog box template identified</td></tr></tbody></table>	Value	Meaning	FR_DIALOGTERM	Indicates the dialog box is closing. The window handle returned by the <u>FindText</u> or <u>ReplaceText</u> function is no longer valid after this bit is set. This flag is set by the system.	FR_DOWN	Sets the direction of searches through a document. If the flag is set, the search direction is down; if the flag is clear, the search direction is up. Initially, this flag specifies the state of the Up and Down buttons; after the user chooses the OK button to close the dialog box, this flag specifies the user's selection.	FR_ENABLEHOOK	Enables the hook function specified in the lpfnHook member of this structure. This flag can be set on input.	FR_ENABLETEMPLATE	Causes the system to use the dialog box template identified
Value	Meaning										
FR_DIALOGTERM	Indicates the dialog box is closing. The window handle returned by the <u>FindText</u> or <u>ReplaceText</u> function is no longer valid after this bit is set. This flag is set by the system.										
FR_DOWN	Sets the direction of searches through a document. If the flag is set, the search direction is down; if the flag is clear, the search direction is up. Initially, this flag specifies the state of the Up and Down buttons; after the user chooses the OK button to close the dialog box, this flag specifies the user's selection.										
FR_ENABLEHOOK	Enables the hook function specified in the lpfnHook member of this structure. This flag can be set on input.										
FR_ENABLETEMPLATE	Causes the system to use the dialog box template identified										

	by the hInstance and lpTemplateName members to display the dialog box. This flag is used only to initialize the dialog box.
FR_ENABLETEMPLATEHANDLE	Indicates that the hInstance member identifies a data block that contains a pre-loaded dialog box template. The system ignores the lpTemplateName member if this flag is specified. This flag can be set on input.
FR_FINDNEXT	Indicates that the application should search for the next occurrence of the string specified by the lpstrFindWhat member. This flag is set by the system.
FR_HIDEMATCHCASE	Hides and disables the Match Case check box. This flag can be set on input.
FR_HIDEWHOLEWORD	Hides and disables the Match Only Whole Word check box. This flag can be set on input.
FR_HIDEUPDOWN	Hides the Up and Down radio buttons that control the direction of searches through a document. This flag can be set on input.
FR_MATCHCASE	Specifies that the search is to be case sensitive. This flag is set when the dialog box is created and may be changed by the system in response to user input.
FR_NOMATCHCASE	Disables the Match Case check box. This flag is used only to initialize the dialog box.
FR_NOUPDOWN	Disables the Up and Down buttons. This flag is used only to initialize the dialog box.
FR_NOWHOLEWORD	Disables the Match Whole Word Only check box. This flag is used only to initialize the dialog box.
FR_REPLACE	Indicates that the application should replace the current occurrence of the string specified in the lpstrFindWhat member with the string specified in the lpstrReplaceWith member. This flag is set by the system.
FR_REPLACEALL	Indicates that the application should replace all occurrences of the string specified in the lpstrFindWhat member with the string specified in the lpstrReplaceWith member. This flag is set by the system.
FR_SHOWHELP	Causes the dialog box to show the Help button. If this flag is specified, the hwndOwner must

		not be NULL. This flag can be set on input.
	FR_WHOLEWORD	Checks the Match Whole Word Only check box. Only whole words that match the search string will be considered. This flag is set when the dialog box is created and may be changed by the system in response to user input.
lpstrFindWhat		Specifies the string to search for. If a string is specified when the dialog box is created, the dialog box will initialize the Find What edit control with this string. If the FR_FINDNEXT flag is set when the dialog box is created, the application should search for an occurrence of this string (using the FR_DOWN, FR_WHOLEWORD, and FR_MATCHCASE flags to further define the direction and type of search). The application must allocate a buffer for the string. This buffer should be at least 80 bytes long. This flag is set when the dialog box is created and may be changed by the system in response to user input.
lpstrReplaceWith		Specifies the replacement string for replace operations. The FindText function ignores this member. The ReplaceText function uses this string to initialize the Replace With edit control. This flag is set when the dialog box is created and may be changed by the system in response to user input.
wFindWhatLen		Specifies the length, in bytes, of the buffer to which the lpstrFindWhat member points. This member is filled on input.
wReplaceWithLen		Specifies the length, in bytes, of the buffer to which the lpstrReplaceWith member points. This member is filled on input.
lCustData		Specifies application-defined data that the system passes to the hook function identified by the lpfnHook member. The system passes a pointer to the <u>CHOOSECOLOR</u> structure in the lParam parameter of the <u>WM_INITDIALOG</u> message; this pointer can be used to retrieve the lCustData member.
lpfnHook		Points to a hook function that processes messages intended for the dialog box. To enable the hook function, an application must specify the FR_ENABLEHOOK flag in the Flags member; otherwise, the system ignores this structure member. The hook function must return zero to pass a message that it didn't process back to the dialog box procedure in COMMDLG.DLL. The hook function must return a nonzero value to prevent the dialog box procedure in COMMDLG.DLL from processing a message it has already processed. This member is filled on input.
lpTemplateName		Points to a null-terminated string that specifies the name of the resource file for the dialog box template that is to be substituted for the dialog box template in COMMDLG.DLL. An application can use the <u>MAKEINTRESOURCE</u> macro for numbered dialog box resources. This member is used only if the Flags member specifies the FR_ENABLETEMPLATE flag; otherwise, this member is ignored. This member is filled on input.

#### Comments

Some members of this structure are filled only when the dialog box is created, some are filled only when the user closes the dialog box, and some have an initialization value that changes when the user closes the dialog box. Whenever a description in the Members section does not specify how the value of a member is assigned, the value is assigned only when the dialog box is created.

#### See Also

FindText, ReplaceText, MAKEINTRESOURCE



## FIXED (3.1)

```
typedef struct tagFIXED { /* fx */
    UINT    fract;
    intvalue;
} FIXED;
```

The FIXED structure contains the integral and fractional parts of a fixed-point real number.

Member	Description
fract	Specifies the fractional part of the number.
value	Specifies the integer part of the number.

### Comments

The FIXED structure is used to describe the elements of the MAT2 and POINTFX structures.

### See Also

GetGlyphOutline, MAT2, POINTFX

## FMS\_GETDRIVEINFO

```
#include <wfext.h>

typedef struct tagFMS_GETDRIVEINFO { /* fmsgdi */
    DWORD dwTotalSpace;
    DWORD dwFreeSpace;
    char  szPath[260];
    char  szVolume[14];
    char  szShare[128];
} FMS_GETDRIVEINFO, FAR *LPFMS_GETDRIVEINFO;
```

The FMS\_GETDRIVEINFO structure contains information about the drive that is selected in the currently active File Manager window.

Member	Description
dwTotalSpace	Specifies the total amount of storage space, in bytes, on the disk associated with the drive.
dwFreeSpace	Specifies the amount of free storage space, in bytes, on the disk associated with the drive.
szPath	Specifies a null-terminated string that contains the path of the current directory.
szVolume	Specifies a null-terminated string that contains the volume label of the disk associated with the drive.
szShare	Specifies a null-terminated string that contains the name of the sharepoint (if the drive is being accessed through a network).

See Also

FMExtensionProc, FM\_GETDRIVEINFO

## FMS\_GETFILESEL

```
#include <wfext.h>

typedef struct tagFMS_GETFILESEL { /* fmsgfs */
    UINT    wTime;
    UINT    wDate;
    DWORD   dwSize;
    BYTE    bAttr;
    char    szName[260];
} FMS_GETFILESEL;
```

The FMS\_GETFILESEL structure contains information about a selected file in File Manager's directory window or Search Results window.

Member	Description
wTime	Specifies the time when the file was created.
wDate	Specifies the date when the file was created.
dwSize	Specifies the size, in bytes, of the file.
bAttr	Specifies the attributes of the file.
szName	Specifies a null-terminated string (an OEM string) that contains the fully-qualified path of the selected file. Before displaying this string, an extension should use the <u>OemToAnsi</u> function to convert the string to a Windows ANSI string. If a string is to be passed to the MS-DOS file system, an extension should not convert it.

See Also

FMEExtensionProc, FM\_GETFILESEL

## FMS\_LOAD

```
#include <wfext.h>

typedef struct tagFMS_LOAD { /* fmsld */
    DWORD dwSize;
    char  szMenuName[MENU_TEXT_LEN];
    HMENU hMenu;
    UINT  wMenuDelta;
} FMS_LOAD;
```

The `FMS_LOAD` structure contains information that File Manager uses to add a custom menu provided by a File Manager extension dynamic-link library (DLL). The structure also provides a delta value that the extension DLL can use to manipulate the custom menu after File Manager has loaded the menu.

Member	Description
<code>dwSize</code>	Specifies the length of the structure, in bytes.
<code>szMenuName</code>	Contains a null-terminated string for a menu item that appears in File Manager's main menu.
<code>hMenu</code>	Identifies the pop-up menu that is added to File Manager's main menu.
<code>wMenuDelta</code>	Specifies the menu-item delta value. To avoid conflicts with its own menu items, File Manager rennumbers the menu-item identifiers in the pop-up menu identified by the <code>hMenu</code> member by adding this delta value to each identifier. An extension DLL that needs to modify a menu item must identify the item to modify by adding the delta value to the menu item's identifier. The value of this member can vary from session to session.

See Also  
`FMEExtensionProc`

## GLOBALENTRY (3.1)

```
#include <toolhelp.h>

typedef struct tagGLOBALENTRY { /* ge */
    DWORD    dwSize;
    DWORD    dwAddress;
    DWORD    dwBlockSize;
    HGLOBAL  hBlock;
    WORD     wcLock;
    WORD     wcPageLock;
    WORD     wFlags;
    BOOL     wHeapPresent;
    HGLOBAL  hOwner;
    WORD     wType;
    WORD     wData;
    DWORD    dwNext;
    DWORD    dwNextAlt;
} GLOBALENTRY;
```

The GLOBALENTRY structure contains information about a memory object on the global heap.

Member	Description																								
dwSize	Specifies the size of the GLOBALENTRY structure, in bytes.																								
dwAddress	Specifies the linear address of the global-memory object.																								
dwBlockSize	Specifies the size of the global-memory object, in bytes.																								
hBlock	Identifies the global-memory object.																								
wcLock	Specifies the lock count. If this value is zero, the memory object is not locked.																								
wcPageLock	Specifies the page lock count. If this value is zero, the memory page is not locked.																								
wFlags	Specifies additional information about the memory object. This member can be the following value:																								
	<table border="0"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>GF_PDB_OWNER</td> <td>The process data block (PDB) for the task is the owner of the memory object.</td> </tr> </tbody> </table>	Value	Meaning	GF_PDB_OWNER	The process data block (PDB) for the task is the owner of the memory object.																				
Value	Meaning																								
GF_PDB_OWNER	The process data block (PDB) for the task is the owner of the memory object.																								
wHeapPresent	Indicates whether a local heap exists within the global-memory object.																								
hOwner	Identifies the owner of the global-memory object.																								
wType	Specifies the memory type of the object. This type can be one of the following values:																								
	<table border="0"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>GT_UNKNOWN</td> <td>The memory type is not known.</td> </tr> <tr> <td>GT_DGROUP</td> <td>The object contains the default data segment and the stack segment.</td> </tr> <tr> <td>GT_DATA</td> <td>The object contains program data. (It may also contain stack and local heap data.)</td> </tr> <tr> <td>GT_CODE</td> <td>The object contains program code. If GT_CODE is specified, the wData member contains the segment number for the code.</td> </tr> <tr> <td>GT_TASK</td> <td>The object contains the task database.</td> </tr> <tr> <td>GT_RESOURCE</td> <td>The object contains the resource type specified in wData.</td> </tr> <tr> <td>GT_MODULE</td> <td>The object contains the module database.</td> </tr> <tr> <td>GT_FREE</td> <td>The object belongs to the free memory pool.</td> </tr> <tr> <td>GT_INTERNAL</td> <td>The object is reserved for internal use by Windows.</td> </tr> <tr> <td>GT_SENTINEL</td> <td>The object is either the first or the last object on the global heap.</td> </tr> <tr> <td>GT_BURGERMASTER</td> <td>The object contains a table that maps selectors to arena handles.</td> </tr> </tbody> </table>	Value	Meaning	GT_UNKNOWN	The memory type is not known.	GT_DGROUP	The object contains the default data segment and the stack segment.	GT_DATA	The object contains program data. (It may also contain stack and local heap data.)	GT_CODE	The object contains program code. If GT_CODE is specified, the wData member contains the segment number for the code.	GT_TASK	The object contains the task database.	GT_RESOURCE	The object contains the resource type specified in wData.	GT_MODULE	The object contains the module database.	GT_FREE	The object belongs to the free memory pool.	GT_INTERNAL	The object is reserved for internal use by Windows.	GT_SENTINEL	The object is either the first or the last object on the global heap.	GT_BURGERMASTER	The object contains a table that maps selectors to arena handles.
Value	Meaning																								
GT_UNKNOWN	The memory type is not known.																								
GT_DGROUP	The object contains the default data segment and the stack segment.																								
GT_DATA	The object contains program data. (It may also contain stack and local heap data.)																								
GT_CODE	The object contains program code. If GT_CODE is specified, the wData member contains the segment number for the code.																								
GT_TASK	The object contains the task database.																								
GT_RESOURCE	The object contains the resource type specified in wData.																								
GT_MODULE	The object contains the module database.																								
GT_FREE	The object belongs to the free memory pool.																								
GT_INTERNAL	The object is reserved for internal use by Windows.																								
GT_SENTINEL	The object is either the first or the last object on the global heap.																								
GT_BURGERMASTER	The object contains a table that maps selectors to arena handles.																								
wData	If the wType member is not GT_CODE or GT_RESOURCE, wData is zero.																								

If wType is GT\_CODE, GT\_DATA, or GT\_DGROUP, wData contains the segment number for the code.

If wType is GT\_RESOURCE, wData specifies the type of resource. The type can be one of the following values:

Value	Meaning
GD_ACCELERATORS	The object contains data from the accelerator table.
GD_BITMAP	The object contains data describing a bitmap. This includes the bitmap color table and the bitmap bits.
GD_CURSOR	The object contains data describing a group of cursors. This includes the height, width, color count, bit count, and ordinal identifier for the cursors.
GD_CURSORCOMPONENT	The object contains data describing a single cursor. This includes bitmap bits and bitmasks for the cursor.
GD_DIALOG	The object contains data describing controls within a dialog box.
GD_ERRRTABLE	The object contains data from the error table.
GD_FONT	The object contains data describing a single font. This data is identical to data in a Windows font file (.FNT).
GD_FONTDIR	The object contains data describing a group of fonts. This includes the number of fonts in the resource and a table of metrics for each of these fonts.
GD_ICON	The object contains data describing a group of icons. This includes the height, width, color count, bit count, and ordinal identifier for the icons.
GD_ICONCOMPONENT	The object contains data describing a single icon. This includes bitmap bits and bitmaps for the icon.
GD_MENU	The object contains menu data for normal and pop-up menu items.
GD_NAMETABLE	The object contains data from the name table.
GD_RCDATA	The object contains data from a user-defined resource.
GD_STRING	The object contains data from the string table.
GD_USERDEFINED	The resource has an unknown resource identifier or is an application-specific named type.

dwNext Reserved for internal use by Windows.

dwNextAlt Reserved for internal use by Windows.

See Also

[GlobalEntryHandle](#), [GlobalEntryModule](#), [GlobalFirst](#), [GlobalNext](#), [GLOBALINFO](#)

## GLOBALINFO (3.1)

```
#include <toolhelp.h>

typedef struct tagGLOBALINFO { /* gi */
    DWORD dwSize;
    WORD  wcItems;
    WORD  wcItemsFree;
    WORD  wcItemsLRU;
} GLOBALINFO;
```

The GLOBALINFO structure contains information about the global heap.

Member	Description
dwSize	Specifies the size of the GLOBALINFO structure, in bytes.
wcItems	Specifies the total number of items on the global heap.
wcItemsFree	Specifies the number of free items on the global heap.
wcItemsLRU	Specifies the number of "least recently used" (LRU) items on the global heap.

See Also

GlobalInfo, GLOBALENTY

## GLYPHMETRICS (3.1)

```
typedef struct tagGLYPHMETRICS { /* gm */
    UINT  gmBlackBoxX;
    UINT  gmBlackBoxY;
    POINT gmptGlyphOrigin;
    int   gmCellIncX;
    int   gmCellIncY;
} GLYPHMETRICS;
```

The GLYPHMETRICS structure contains information about the placement and orientation of a glyph in a character cell.

Member	Description
gmBlackBoxX	Specifies the width of the smallest rectangle that completely encloses the glyph (its "black box").
gmBlackBoxY	Specifies the height of the smallest rectangle that completely encloses the glyph (its "black box").
gmptGlyphOrigin	Specifies the x- and y-coordinates of the upper-left corner of the smallest rectangle that completely encloses the glyph.
gmCellIncX	Specifies the horizontal distance from the origin of the current character cell to the origin of the next character cell.
gmCellIncY	Specifies the vertical distance from the origin of the current character cell to the origin of the next character cell.

### Comments

Values in the GLYPHMETRICS structure are specified in logical units.

### See Also

[GetGlyphOutline](#), [POINT](#)



## HANDLETABLE (2.x)

```
typedef struct tagHANDLETABLE { /* ht */
    HGDIOBJ objectHandle[1];
} HANDLETABLE;
```

The HANDLETABLE structure is an array of handles, each of which identifies a graphics device interface (GDI) object.

Member	Description
objectHandle	Contains an array of handles.

See Also

[EnumMetaFile](#), [PlayMetaFileRecord](#)

## HARDWAREHOOKSTRUCT (3.1)

```
typedef struct tagHARDWAREHOOKSTRUCT { /* hhs */
    HWND    hWnd;
    UINT    wMessage;
    WPARAM  wParam;
    LPARAM  lParam;
} HARDWAREHOOKSTRUCT;
```

The HARDWAREHOOKSTRUCT contains information about a hardware message placed in the system message queue.

Member	Description
hWnd	Identifies the window that will receive the message.
wMessage	Specifies the message identifier.
wParam	Specifies additional information about the message. The exact meaning depends on the wMessage parameter.
lParam	Specifies additional information about the message. The exact meaning depends on the wMessage parameter.

## HELPWININFO (3.1)

```
typedef struct { /* hi */
    int  wStructSize;
    int  x;
    int  y;
    int  dx;
    int  dy;
    int  wMax;
    char rgchMember[2];
} HELPWININFO;
```

The HELPWININFO structure contains the size and position of a secondary help window. An application can set this size by calling the [WinHelp](#) function with the HELP\_SETWINPOS value.

Member	Description
wStructSize	Specifies the size of the HELPWININFO structure.
x	Specifies the x-coordinate of the upper-left corner of the window.
y	Specifies the y-coordinate of the upper-left corner of the window.
dx	Specifies the width of the window.
dy	Specifies the height of the window.
wMax	Specifies whether the window should be maximized or set to the given position and dimensions. If this value is 1, the window is maximized. If it is zero, the size and position of the window are determined by the x, y, dx, and dy members.
rgchMember	Specifies the name of the window.

### Comments

Microsoft Windows Help divides the display into 1024 units in both the x- and y-directions. To create a secondary window that fills the upper-left quadrant of the display, for example, an application would specify zero for the x and y members and 512 for the dx and dy members.

### See Also

[WinHelp](#)

## HSZPAIR (3.1)

```
#include <ddeml.h>

typedef struct tagHSZPAIR {    /* hp */
    HSZ hszSvc;
    HSZ hszTopic;
} HSZPAIR;
```

The HSZPAIR structure contains a dynamic data exchange (DDE) service name and topic name. A DDE server application can use this structure during an XTYP\_WILDCONNECT transaction to enumerate the service/topic name pairs that it supports.

Member	Description
hszSvc	Identifies a service name.
hszTopic	Identifies a topic name.

See Also

XTYP\_WILDCONNECT

## KERNINGPAIR (3.1)

```
typedef struct tagKERNINGPAIR {    /* kp */
    WORD wFirst;
    WORD wSecond;
    int iKernAmount;
} KERNINGPAIR;
```

The KERNINGPAIR structure defines a kerning pair.

Member	Description
wFirst	Specifies the character code for the first character in the kerning pair.
wSecond	Specifies the character code for the second character in the kerning pair.
iKernAmount	Specifies the amount that this pair will be kerned if they appear side by side in the same font and size. This value is typically negative, because pair-kerning usually results in two characters being set more tightly than normal. The value is given in logical units--that is, it depends on the current mapping mode.

See Also

[GetKerningPairs](#)

## LOCALENTRY (3.1)

```
#include <toolhelp.h>

typedef struct tagLOCALENTRY { /* le */
    DWORD    dwSize;
    HLOCAL   hHandle;
    WORD     wAddress;
    WORD     wSize;
    WORD     wFlags;
    WORD     wcLock;
    WORD     wType;
    WORD     hHeap;
    WORD     wHeapType;
    WORD     wNext;
} LOCALENTRY;
```

The LOCALENTRY structure contains information about a memory object on the local heap.

Member	Description																														
dwSize	Specifies the size of the LOCALENTRY structure, in bytes.																														
hHandle	Identifies the local-memory object.																														
wAddress	Specifies the address of the local-memory object.																														
wSize	Specifies the size of the local-memory object, in bytes.																														
wFlags	Specifies whether the memory object is fixed, free, or movable. This member can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>LF_FIXED</td><td>The object resides in a fixed memory location.</td></tr><tr><td>LF_FREE</td><td>The object is part of the free memory pool.</td></tr><tr><td>LF_MOVEABLE</td><td>The object can be moved in order to compact memory.</td></tr></tbody></table>	Value	Meaning	LF_FIXED	The object resides in a fixed memory location.	LF_FREE	The object is part of the free memory pool.	LF_MOVEABLE	The object can be moved in order to compact memory.																						
Value	Meaning																														
LF_FIXED	The object resides in a fixed memory location.																														
LF_FREE	The object is part of the free memory pool.																														
LF_MOVEABLE	The object can be moved in order to compact memory.																														
wcLock	Specifies the lock count. If this value is zero, the memory object is not locked.																														
wType	Specifies the content of the memory object. This member can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>LT_FREE</td><td>The object belongs to the free memory pool.</td></tr><tr><td>LT_GDI_BITMAP</td><td>The object contains a bitmap header.</td></tr><tr><td>LT_GDI_BRUSH</td><td>The object contains a brush.</td></tr><tr><td>LT_GDI_DC</td><td>The object contains a device context.</td></tr><tr><td>LT_GDI_DISABLED_DC</td><td>The object is reserved for internal use by Windows.</td></tr><tr><td>LT_GDI_FONT</td><td>The object contains a font header.</td></tr><tr><td>LT_GDI_MAX</td><td>The object is reserved for internal use by Windows.</td></tr><tr><td>LT_GDI_METADC</td><td>The object contains a metafile device context.</td></tr><tr><td>LT_GDI_METAFILE</td><td>The object contains a metafile header.</td></tr><tr><td>LT_GDI_PALETTE</td><td>The object contains a palette.</td></tr><tr><td>LT_GDI_PEN</td><td>The object contains a pen.</td></tr><tr><td>LT_GDI_RGN</td><td>The object contains a region.</td></tr><tr><td>LT_NORMAL</td><td>The object is reserved for internal use by Windows.</td></tr><tr><td>LT_USER_ATOMS</td><td>The object contains an atom structure.</td></tr></tbody></table>	Value	Meaning	LT_FREE	The object belongs to the free memory pool.	LT_GDI_BITMAP	The object contains a bitmap header.	LT_GDI_BRUSH	The object contains a brush.	LT_GDI_DC	The object contains a device context.	LT_GDI_DISABLED_DC	The object is reserved for internal use by Windows.	LT_GDI_FONT	The object contains a font header.	LT_GDI_MAX	The object is reserved for internal use by Windows.	LT_GDI_METADC	The object contains a metafile device context.	LT_GDI_METAFILE	The object contains a metafile header.	LT_GDI_PALETTE	The object contains a palette.	LT_GDI_PEN	The object contains a pen.	LT_GDI_RGN	The object contains a region.	LT_NORMAL	The object is reserved for internal use by Windows.	LT_USER_ATOMS	The object contains an atom structure.
Value	Meaning																														
LT_FREE	The object belongs to the free memory pool.																														
LT_GDI_BITMAP	The object contains a bitmap header.																														
LT_GDI_BRUSH	The object contains a brush.																														
LT_GDI_DC	The object contains a device context.																														
LT_GDI_DISABLED_DC	The object is reserved for internal use by Windows.																														
LT_GDI_FONT	The object contains a font header.																														
LT_GDI_MAX	The object is reserved for internal use by Windows.																														
LT_GDI_METADC	The object contains a metafile device context.																														
LT_GDI_METAFILE	The object contains a metafile header.																														
LT_GDI_PALETTE	The object contains a palette.																														
LT_GDI_PEN	The object contains a pen.																														
LT_GDI_RGN	The object contains a region.																														
LT_NORMAL	The object is reserved for internal use by Windows.																														
LT_USER_ATOMS	The object contains an atom structure.																														

LT_USER_BWL	The object is reserved for internal use by Windows.
LT_USER_CBOX	The object contains a combo-box structure.
LT_USER_CHECKPOINT	The object is reserved for internal use by Windows.
LT_USER_CLASS	The object contains a class structure.
LT_USER_CLIP	The object is reserved for internal use by Windows.
LT_USER_DCE	The object is reserved for internal use by Windows.
LT_USER_ED	The object contains an edit-control structure.
LT_USER_HANDLETABLE	The object is reserved for internal use by Windows.
LT_USER_HOOKLIST	The object is reserved for internal use by Windows.
LT_USER_HOTKEYLIST	The object is reserved for internal use by Windows.
LT_USER_LBIV	The object contains a list-box structure.
LT_USER_LOCKINPUTSTATE	The object is reserved for internal use by Windows.
LT_USER_MENU	The object contains a menu structure.
LT_USER_MISC	The object is reserved for internal use by Windows.
LT_USER_MWP	The object is reserved for internal use by Windows.
LT_USER_OWNERDRAW	The object is reserved for internal use by Windows.
LT_USER_PALETTE	The object is reserved for internal use by Windows.
LT_USER_POPUPMENU	The object is reserved for internal use by Windows.
LT_USER_PROP	The object contains a window-property structure.
LT_USER_SPB	The object is reserved for internal use by Windows.
LT_USER_STRING	The object is reserved for internal use by Windows.
LT_USER_USERSEEUSERDOALLOC	The object is reserved for internal use by Windows.
LT_USER_WND	The object contains a window structure.

hHeap	Identifies the local-memory heap.
wHeapType	Specifies the type of local heap. This type can be one of the following values:
	Value                      Meaning
	NORMAL_HEAP              The heap is the default heap.
	USER_HEAP                 The heap is used by the USER module.
	GDI_HEAP                  The heap is used by the GDI module.
wNext	Specifies the next entry in the local heap. This member is reserved for internal use by Windows.

#### Comments

The wType values are for informational purposes only. Microsoft reserves the right to change or delete these tags at any time. Applications should never directly change items on the system heaps, as this

information will change in future versions. The wType values for the USER module are included only in the debugging versions of USER.EXE.

See Also

LocalFirst, LocalNext, LOCALINFO



## LOCALINFO (toolhelp 3.1)

```
#include <toolhelp.h>
```

```
typedef struct tagLOCALINFO { /* li */  
    DWORD   dwSize;  
    WORD    wcItems;  
} LOCALINFO;
```

The LOCALINFO structure contains information about the local heap.

Member	Description
dwSize	Specifies the size of the LOCALINFO structure, in bytes.
wcItems	Specifies the total number of items on the local heap.

See Also

LocalInfo, LOCALENTRY

## LOGBRUSH (2.x)

```
typedef struct tagLOGBRUSH {      /* lb */
    UINT lbStyle;
    COLORREF lbColor;
    int lbHatch;
} LOGBRUSH;
```

The LOGBRUSH structure defines the style, color, and pattern of a physical brush to be created by using the CreateBrushIndirect function.

Member	Description														
lbStyle	Specifies the brush style. This member can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>BS_DIBPATTERN</td><td>Specifies a pattern brush defined by a device-independent bitmap (DIB) specification.</td></tr><tr><td>BS_HATCHED</td><td>Specifies a hatched brush.</td></tr><tr><td>BS_HOLLOW</td><td>Specifies a hollow brush.</td></tr><tr><td>BS_PATTERN</td><td>Specifies a pattern brush defined by a memory bitmap.</td></tr><tr><td>BS_NULL</td><td>Equivalent to BS_HOLLOW.</td></tr><tr><td>BS_SOLID</td><td>Specifies a solid brush.</td></tr></tbody></table>	Value	Meaning	BS_DIBPATTERN	Specifies a pattern brush defined by a device-independent bitmap (DIB) specification.	BS_HATCHED	Specifies a hatched brush.	BS_HOLLOW	Specifies a hollow brush.	BS_PATTERN	Specifies a pattern brush defined by a memory bitmap.	BS_NULL	Equivalent to BS_HOLLOW.	BS_SOLID	Specifies a solid brush.
Value	Meaning														
BS_DIBPATTERN	Specifies a pattern brush defined by a device-independent bitmap (DIB) specification.														
BS_HATCHED	Specifies a hatched brush.														
BS_HOLLOW	Specifies a hollow brush.														
BS_PATTERN	Specifies a pattern brush defined by a memory bitmap.														
BS_NULL	Equivalent to BS_HOLLOW.														
BS_SOLID	Specifies a solid brush.														
lbColor	Specifies the color in which the brush is to be drawn. If the lbStyle member is the BS_HOLLOW or BS_PATTERN value, lbColor is ignored. If lbStyle is the BS_DIBPATTERN value, the low-order word of lbColor specifies whether the bmiColors members of the BITMAPINFO structure contain explicit RGB values or indexes into the currently realized <u>logical palette</u> . The lbColor member <u>must</u> be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>DIB_PAL_COLORS</td><td>Color table consists of an array of 16-bit indexes into the currently realized logical palette.</td></tr><tr><td>DIB_RGB_COLORS</td><td>Color table contains literal <u>RGB</u> values.</td></tr></tbody></table>	Value	Meaning	DIB_PAL_COLORS	Color table consists of an array of 16-bit indexes into the currently realized logical palette.	DIB_RGB_COLORS	Color table contains literal <u>RGB</u> values.								
Value	Meaning														
DIB_PAL_COLORS	Color table consists of an array of 16-bit indexes into the currently realized logical palette.														
DIB_RGB_COLORS	Color table contains literal <u>RGB</u> values.														
lbHatch	Specifies a hatch style. The meaning depends on the brush style. If the lbStyle member is the BS_DIBPATTERN style, the lbHatch member contains a handle to a packed DIB. To obtain this handle, an application calls the <u>GlobalAlloc</u> function to allocate a global memory object and then fills the memory with the <u>packed DIB</u> . A packed DIB consists of a BITMAPINFO structure immediately followed by the array of bytes which define the <u>pixels of the bitmap</u> . If the lbStyle member is the BS_HATCHED style, the lbHatch member specifies the orientation of the lines used to create the hatch. This member can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>HS_BDIAGONAL</td><td>45-degree upward hatch (left to right)</td></tr><tr><td>HS_CROSS</td><td>Horizontal and vertical cross-hatch</td></tr><tr><td>HS_DIAGCROSS</td><td>45-degree cross-hatch</td></tr><tr><td>HS_FDIAGONAL</td><td>45-degree downward hatch (left to right)</td></tr><tr><td>HS_HORIZONTAL</td><td>Horizontal hatch</td></tr><tr><td>HS_VERTICAL</td><td>Vertical hatch</td></tr></tbody></table> If the lbStyle member is the BS_PATTERN style, lbHatch must be a handle to the bitmap that defines the pattern. If the lbStyle member is the BS_SOLID or the BS_HOLLOW style, lbHatch is ignored.	Value	Meaning	HS_BDIAGONAL	45-degree upward hatch (left to right)	HS_CROSS	Horizontal and vertical cross-hatch	HS_DIAGCROSS	45-degree cross-hatch	HS_FDIAGONAL	45-degree downward hatch (left to right)	HS_HORIZONTAL	Horizontal hatch	HS_VERTICAL	Vertical hatch
Value	Meaning														
HS_BDIAGONAL	45-degree upward hatch (left to right)														
HS_CROSS	Horizontal and vertical cross-hatch														
HS_DIAGCROSS	45-degree cross-hatch														
HS_FDIAGONAL	45-degree downward hatch (left to right)														
HS_HORIZONTAL	Horizontal hatch														
HS_VERTICAL	Vertical hatch														

See Also

BITMAPINFO, CreateBrushIndirect, CreateBrushIndirect, GlobalAlloc

## LOGFONT (2.x)

```
typedef struct tagLOGFONT { /* lf */
    int    lfHeight;
    int    lfWidth;
    int    lfEscapement;
    int    lfOrientation;
    int    lfWeight;
    BYTE   lfItalic;
    BYTE   lfUnderline;
    BYTE   lfStrikeOut;
    BYTE   lfCharSet;
    BYTE   lfOutPrecision;
    BYTE   lfClipPrecision;
    BYTE   lfQuality;
    BYTE   lfPitchAndFamily;
    BYTE   lfFaceName[LF_FACESIZE];
} LOGFONT;
```

The LOGFONT structure defines the attributes of a font, a drawing object used to write text on a display surface.

Member	Description
lfHeight	Specifies the desired height, in logical units, for the font. If this value is greater than zero, it specifies the cell height of the font. If it is less than zero, it specifies the character height of the font. (Character height is the cell height minus the internal leading. Applications that specify font height in points typically use a negative number for this member.) If this value is zero, the font mapper uses a default height. The font mapper chooses the largest physical font that does not exceed the requested size (or the smallest font, if all the fonts exceed the requested size). The absolute value of the lfHeight member must not exceed 16,384 after it is converted to device units.
lfWidth	Specifies the average width, in logical units, of characters in the font. If this value is zero, the font mapper chooses a reasonable default width for the specified font height. (The default width is chosen by matching the aspect ratio of the device against the digitization aspect ratio of the available fonts. The closest match is determined by the absolute value of the difference.) The widths of characters in TrueType fonts are scaled by a factor of this member divided by the width of the characters in the physical font (as specified by the tmAveCharWidth member of the <a href="#">TEXTMETRIC</a> structure).
lfEscapement	Specifies the angle, in tenths of degrees, between the base line of a character and the x-axis. The angle is measured in a counterclockwise direction from the x-axis for left-handed coordinate systems (that is, MM_TEXT, in which the y direction is down) and in a clockwise direction from the x-axis for right-handed coordinate systems (in which the y direction is up).
lfOrientation	Specifies the orientation of the characters. This value is ignored.
lfWeight	Specifies the font weight. This member can be one of the following values:

Constant	Value
FW_DONTCARE	0
FW_THIN	100
FW_EXTRALIGHT	200
FW_ULTRALIGHT	200
FW_LIGHT	300
FW_NORMAL	400
FW_REGULAR	400
FW_MEDIUM	500
FW_SEMIBOLD	600
FW_DEMIBOLD	600
FW_BOLD	700
FW_EXTRABOLD	800

FW_ULTRABOLD	800
FW_BLACK	900
FW_HEAVY	900

The actual appearance of the font depends on the type face. Some fonts have only FW\_NORMAL, FW\_REGULAR, and FW\_BOLD weights. If FW\_DONTCARE is specified, a default weight is used.

lfiItalic  
lfiUnderline  
lfiStrikeOut  
lfiCharSet

Specifies an italic font if nonzero.

Specifies an underlined font if nonzero.

Specifies a strikeout font if nonzero.

Specifies the character set of the font. The following values are predefined:

Constant	Value
ANSI_CHARSET	0
DEFAULT_CHARSET	1
SYMBOL_CHARSET	2
SHIFTJIS_CHARSET	128
OEM_CHARSET	255

The DEFAULT\_CHARSET value is not used by the font mapper. An application can use this value to allow the name and size of a font to fully describe the logical font. If the specified font name does not exist, a font from any character set can be substituted for the specified font; applications should use the DEFAULT\_CHARSET value sparingly to avoid unexpected results.

The OEM character set is system-dependent.

Fonts with other character sets may exist in the system. If an application uses a font with an unknown character set, it should not attempt to translate or interpret strings that are to be rendered with that font.

lfiOutPrecision

Specifies the desired output precision. The output precision defines how closely the output must match the height, width, character orientation, escapement, and pitch of the requested font. This member can be one of the following values:

OUT_CHARACTER_PRECIS	OUT_STRING_PRECIS
OUT_DEFAULT_PRECIS	OUT_STROKE_PRECIS
OUT_DEVICE_PRECIS	OUT_TT_PRECIS
OUT_RASTER_PRECIS	OUT_TT_ONLY_PRECIS

Applications can use the values OUT\_DEVICE\_PRECIS, OUT\_RASTER\_PRECIS, and OUT\_TT\_PRECIS to control how the font mapper chooses a font when the system contains more than one font with a given name. For example, if a system contains a font named "Symbol" in raster and TrueType form, specifying OUT\_TT\_PRECIS would force the font mapper to choose the TrueType version. (Specifying OUT\_TT\_PRECIS forces the font mapper to choose a TrueType font whenever the specified font name matches a device or raster font, even when there is no TrueType font with the same name.)

An application can use TrueType fonts exclusively by specifying OUT\_TT\_ONLY\_PRECIS. When this value is specified, the system chooses a TrueType font even when the name specified in the lfiFaceName member matches a raster or vector font.

lfiClipPrecision

Specifies the desired clipping precision. The clipping precision defines how to clip characters that are partially outside the clipping region. This member can be any one of the following values:

CLIP_CHARACTER_PRECIS	CLIP_MASK
CLIP_DEFAULT_PRECIS	CLIP_STROKE_PRECIS
CLIP_EMBEDDED	CLIP_TT_ALWAYS
CLIP_LH_ANGLES	

To use an embedded read-only font, applications must specify the CLIP\_EMBEDDED value.

To achieve consistent rotation of device, TrueType, and vector fonts, an application can use the OR operator to combine the CLIP\_LH\_ANGLES value

with any of the other `IfClipPrecision` values. If the `CLIP_LH_ANGLES` bit is set, the rotation for all fonts is dependent on whether the orientation of the coordinate system is left-handed or right-handed. If `CLIP_LH_ANGLES` is not set, device fonts always rotate counter-clockwise, but the rotation of other fonts is dependent on the orientation of the coordinate system. (For more information about the orientation of coordinate systems, see the description of the `IfEscapement` member.)

`IfQuality`

Specifies the output quality of the font, which defines how carefully the graphics device interface (GDI) must attempt to match the logical-font attributes to those of an actual physical font. This member can be one of the following values:

Value	Meaning
<code>DEFAULT_QUALITY</code>	Appearance of the font does not matter.
<code>DRAFT_QUALITY</code>	Appearance of the font is less important than when the <code>PROOF_QUALITY</code> value is used. For GDI raster fonts, scaling is enabled. Bold, italic, underline, and strikeout fonts are synthesized if necessary.
<code>PROOF_QUALITY</code>	Character quality of the font is more important than exact matching of the logical-font attributes. For GDI raster fonts, scaling is disabled and the font closest in size is chosen. Bold, italic, underline, and strikeout fonts are synthesized if necessary.

`IfPitchAndFamily`

Specifies the pitch and family of the font. The two low-order bits, which specify the pitch of the font, can be one of the following values:

<code>DEFAULT_PITCH</code>	<code>VARIABLE_PITCH</code>
<code>FIXED_PITCH</code>	

The four high-order bits of the member, which specify the font family, can be one of the following values:

Value	Meaning
<code>FF_DECORATIVE</code>	Novelty fonts. Old English is an example.
<code>FF_DONTCARE</code>	Don't care or don't know.
<code>FF_MODERN</code>	Fonts with constant stroke width, with or without serifs. Pica, Elite, and Courier New are examples.
<code>FF_ROMAN</code>	Fonts with variable stroke width and with serifs. Times New Roman and New Century Schoolbook are examples.
<code>FF_SCRIPT</code>	Fonts designed to look like handwriting. Script and Cursive are examples.
<code>FF_SWISS</code>	Fonts with variable stroke width and without serifs. <u>MS</u> Sans Serif is an example.

An application can specify a value for the `IfPitchAndFamily` member by using the Boolean OR operator to join a pitch constant with a family constant.

Font families describe the look of a font in a general way. They are intended for specifying fonts when the exact typeface desired is not available.

`IfFaceName`

Specifies the typeface name of the font. The length of this string must not exceed `LF_FACESIZE - 1`. The `EnumFontFamilies` function can be used to enumerate the typeface names of all currently available fonts. If `IfFaceName` is `NULL`, GDI uses a device-dependent typeface.

Comments

Applications can use the default settings for most of these members when creating a logical font. The members that should always be given specific values are `IfHeight` and `IfFaceName`. If `IfHeight` and `IfFaceName` are not set by the application, the logical font that is created is device-dependent.

See Also

[CreateFontIndirect](#), [EnumFontFamilies](#)

## LOGPALETTE (3.0)

```
typedef struct tagLOGPALETTE { /* lgpl */
    WORD    palVersion;
    WORD    palNumEntries;
    PALETTEENTRY palPalEntry[1];
} LOGPALETTE;
```

The LOGPALETTE structure defines a logical color palette.

Member	Description
palVersion	Specifies the Windows version number for the structure. This value should be 0x300 for Windows 3.0 and later.
palNumEntries	Specifies the number of palette color entries.
palPalEntry	Specifies an array of <u>PALETTEENTRY</u> structures that define the color and usage of each entry in the logical palette.

### Comments

The colors in the palette entry table should appear in order of importance, because entries earlier in the logical palette are most likely to be placed in the system palette.

This structure is passed as a parameter to the CreatePalette function.

### See Also

CreatePalette, PALETTEENTRY

## LOGPEN (2.x)

```
typedef struct tagLOGPEN { /* lgpn */
    UINT lopnStyle;
    POINT  lopnWidth;
    COLORREF lopnColor;
} LOGPEN;
```

The LOGPEN structure defines the style, width, and color of a pen, a drawing object used to draw lines and borders. The [CreatePenIndirect](#) function uses the LOGPEN structure.

Member	Description																
lopnStyle	Specifies the pen type. This member can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>PS_SOLID</td><td>Creates a solid pen.</td></tr><tr><td>PS_DASH</td><td>Creates a dashed pen. (Valid only when the pen width is 1.)</td></tr><tr><td>PS_DOT</td><td>Creates a dotted pen. (Valid only when the pen width is 1.)</td></tr><tr><td>PS_DASHDOT</td><td>Creates a pen with alternating dashes and dots. (Valid only when the pen width is 1.)</td></tr><tr><td>PS_DASHDOTDOT</td><td>Creates a pen with alternating dashes and double dots. (Valid only when the pen width is 1.)</td></tr><tr><td>PS_NULL</td><td>Creates a null pen.</td></tr><tr><td>PS_INSIDEFRAME</td><td>Creates a pen that draws a line inside the frame of closed shapes produced by graphics device interface (GDI) output functions that specify a bounding rectangle (for example, the <a href="#">Ellipse</a>, <a href="#">Rectangle</a>, <a href="#">RoundRect</a>, <a href="#">Pie</a>, and <a href="#">Chord</a> functions). When this style is used with GDI output functions that do not specify a bounding rectangle (for example, the <a href="#">LineTo</a> function), the drawing area of the pen is not limited by a frame.</td></tr></tbody></table>	Value	Meaning	PS_SOLID	Creates a solid pen.	PS_DASH	Creates a dashed pen. (Valid only when the pen width is 1.)	PS_DOT	Creates a dotted pen. (Valid only when the pen width is 1.)	PS_DASHDOT	Creates a pen with alternating dashes and dots. (Valid only when the pen width is 1.)	PS_DASHDOTDOT	Creates a pen with alternating dashes and double dots. (Valid only when the pen width is 1.)	PS_NULL	Creates a null pen.	PS_INSIDEFRAME	Creates a pen that draws a line inside the frame of closed shapes produced by graphics device interface (GDI) output functions that specify a bounding rectangle (for example, the <a href="#">Ellipse</a> , <a href="#">Rectangle</a> , <a href="#">RoundRect</a> , <a href="#">Pie</a> , and <a href="#">Chord</a> functions). When this style is used with GDI output functions that do not specify a bounding rectangle (for example, the <a href="#">LineTo</a> function), the drawing area of the pen is not limited by a frame.
Value	Meaning																
PS_SOLID	Creates a solid pen.																
PS_DASH	Creates a dashed pen. (Valid only when the pen width is 1.)																
PS_DOT	Creates a dotted pen. (Valid only when the pen width is 1.)																
PS_DASHDOT	Creates a pen with alternating dashes and dots. (Valid only when the pen width is 1.)																
PS_DASHDOTDOT	Creates a pen with alternating dashes and double dots. (Valid only when the pen width is 1.)																
PS_NULL	Creates a null pen.																
PS_INSIDEFRAME	Creates a pen that draws a line inside the frame of closed shapes produced by graphics device interface (GDI) output functions that specify a bounding rectangle (for example, the <a href="#">Ellipse</a> , <a href="#">Rectangle</a> , <a href="#">RoundRect</a> , <a href="#">Pie</a> , and <a href="#">Chord</a> functions). When this style is used with GDI output functions that do not specify a bounding rectangle (for example, the <a href="#">LineTo</a> function), the drawing area of the pen is not limited by a frame.																
lopnWidth	Specifies the pen width, in logical units. If the lopnWidth member is zero, the pen is one pixel wide on raster devices regardless of the current mapping mode.																
lopnColor	Specifies the pen color.																

### Comments

The y value in the [POINT](#) structure for the lopnWidth member is not used.

### See Also

[CreatePenIndirect](#), [POINT](#)

The following shows how the various pens appear when used to draw a rectangle:

■

## MAT2 (3.1)

```
typedef struct tagMAT2 { /* mat2 */
    FIXED eM11;
    FIXED eM12;
    FIXED eM21;
    FIXED eM22;
} MAT2;
```

The MAT2 structure contains the values for a transformation matrix.

Member	Description
eM11	Specifies a fixed-point value for the M11 component of a 2-by-2 transformation matrix.
eM12	Specifies a fixed-point value for the M12 component of a 2-by-2 transformation matrix.
eM21	Specifies a fixed-point value for the M21 component of a 2-by-2 transformation matrix.
eM22	Specifies a fixed-point value for the M22 component of a 2-by-2 transformation matrix.

### Comments

The identity matrix produces a transformation in which the transformed graphical object is identical to the source object. In the identity matrix, the value of eM11 is 1, the value of eM12 is zero, the value of eM21 is zero, and the value of eM22 is 1.

### See Also

[GetGlyphOutline](#), [FIXED](#)



## MDICREATESTRUCT (3.0)

```
typedef struct tagMDICREATESTRUCT { /* mdic */
    LPCSTR    szClass;
    LPCSTR    szTitle;
    HINSTANCE hOwner;
    int       x;
    int       y;
    int       cx;
    int       cy;
    DWORD style;
    LPARAM    lParam;
} MDICREATESTRUCT;
```

The MDICREATESTRUCT structure contains information about the class, title, owner, location, and size of a multiple document interface (MDI) child window.

Member	Description										
szClass	Contains a long pointer to the application-defined class of the MDI child window.										
szTitle	Contains a long pointer to the window title of the MDI child window.										
hOwner	Identifies the instance handle of the application creating the MDI child window.										
x	Specifies the initial position of the left side of the MDI child window. If this member is set to CW_USEDEFAULT, the MDI child window is assigned a default horizontal position.										
y	Specifies the initial position of the top edge of the MDI child window. If this member is set to CW_USEDEFAULT, the MDI child window is assigned a default vertical position.										
cx	Specifies the initial width of the MDI child window. If this member is set to CW_USEDEFAULT, the MDI child window is assigned a default width.										
cy	Specifies the initial height of the MDI child window. If this member is set to CW_USEDEFAULT, the MDI child window is assigned a default height.										
style	Specifies additional styles for the MDI child window. If the MDI client window was created with the MDIS_ALLCHILDSTYLES window style, the style member may be any combination of the window styles documented with the <a href="#">CreateWindow</a> function. Otherwise, it may be one or more of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>WS_MINIMIZE</td><td>MDI child window is created in a minimized state.</td></tr><tr><td>WS_MAXIMIZE</td><td>MDI child window is created in a maximized state.</td></tr><tr><td>WS_HSCROLL</td><td>MDI child window is created with a horizontal scroll bar.</td></tr><tr><td>WS_VSCROLL</td><td>MDI child window is created with a vertical scroll bar.</td></tr></tbody></table>	Value	Meaning	WS_MINIMIZE	MDI child window is created in a minimized state.	WS_MAXIMIZE	MDI child window is created in a maximized state.	WS_HSCROLL	MDI child window is created with a horizontal scroll bar.	WS_VSCROLL	MDI child window is created with a vertical scroll bar.
Value	Meaning										
WS_MINIMIZE	MDI child window is created in a minimized state.										
WS_MAXIMIZE	MDI child window is created in a maximized state.										
WS_HSCROLL	MDI child window is created with a horizontal scroll bar.										
WS_VSCROLL	MDI child window is created with a vertical scroll bar.										
lParam	Specifies an application-defined 32-bit value.										

### Comments

When the MDI child window is created, Windows sends the [WM\\_CREATE](#) message to the window. The lParam parameter of the WM\_CREATE message contains a pointer to a [CREATESTRUCT](#) structure. The lpCreateParams member of [CREATESTRUCT](#) contains a pointer to the [MDICREATESTRUCT](#) structure passed with the [WM\\_MDICREATE](#) message that created the MDI child window.

### See Also

[CREATESTRUCT](#)

## MEASUREITEMSTRUCT (3.0)

```
typedef struct tagMEASUREITEMSTRUCT {    /* mi */
    UINT    CtlType;
    UINT    CtlID;
    UINT    itemID;
    UINT    itemWidth;
    UINT    itemHeight;
    DWORD   itemData;
} MEASUREITEMSTRUCT;
```

The MEASUREITEMSTRUCT structure informs Windows of the dimensions of an owner-drawn control. This allows Windows to process user interaction with the control correctly. The owner of an owner-drawn control receives a pointer to this structure as the lParam parameter of an WM\_MEASUREITEM message. The owner-drawn control sends this message to its owner window when the control is created. The owner then fills in the appropriate members in the structure for the control and returns. This structure is common to all owner-drawn controls.

Member	Description	
CtlType	Specifies the control type. The values for control types are as follows:	
	Value	Meaning
	ODT_BUTTON	Owner-drawn button
	ODT_COMBOBOX	Owner-drawn combo box
	ODT_LISTBOX	Owner-drawn list box
ODT_MENU	Owner-drawn menu	
CtlID	Specifies the control identifier for a combo box, list box, or button. This member is not used for a menu.	
itemID	Specifies the menu-item identifier for a menu or the list-box item identifier for a variable-height combo box or list box. This member is not used for a fixed-height combo box or list box or for a button.	
itemWidth	Specifies the width of a menu item. The owner of the owner-drawn menu item must fill this member before returning from the message.	
itemHeight	Specifies the height of an individual item in a list box or a menu. Before returning from the message, the owner of the owner-drawn combo box, list box, or menu item must fill out this member. The maximum height of a list box item is 255.	
itemData	Contains the value that was passed to the combo box or list box in the lParam parameter of one of the following messages:  <u>CB_ADDSTRING</u> <u>CB_INSERTSTRING</u> <u>LB_ADDSTRING</u> <u>LB_INSERTSTRING</u>	

### Comments

Failure to fill out the proper members in the MEASUREITEMSTRUCT structure will cause improper operation of the control.

### See Also

WM\_MEASUREITEM

## MEMMANINFO (toolhelp 3.1)

```
#include <toolhelp.h>

typedef struct tagMEMMANINFO { /* mmi */
    DWORD dwSize;
    DWORD dwLargestFreeBlock;
    DWORD dwMaxPagesAvailable;
    DWORD dwMaxPagesLockable;
    DWORD dwTotalLinearSpace;
    DWORD dwTotalUnlockedPages;
    DWORD dwFreePages;
    DWORD dwTotalPages;
    DWORD dwFreeLinearSpace;
    DWORD dwSwapFilePages;
    WORD wPageSize;
} MEMMANINFO;
```

The MEMMANINFO structure contains information about the status and performance of the virtual-memory manager. If the memory manager is running in standard mode, the only valid member of this structure is the dwLargestFreeBlock member.

Member	Description
dwSize	Specifies the size of the MEMMANINFO structure, in bytes.
dwLargestFreeBlock	Specifies the largest free block of contiguous linear memory in the system, in bytes.
dwMaxPagesAvailable	Specifies the maximum number of pages that could be allocated in the system (the value of dwLargestFreeBlock divided by the value of wPageSize).
dwMaxPagesLockable	Specifies the maximum number of pages that could be allocated and locked.
dwTotalLinearSpace	Specifies the size of the total linear address space, in pages.
dwTotalUnlockedPages	Specifies the number of unlocked pages in the system. This value includes free pages.
dwFreePages	Specifies the number of pages that are not in use.
dwTotalPages	Specifies the total number of pages the virtual-memory manager manages. This value includes free, locked, and unlocked pages.
dwFreeLinearSpace	Specifies the amount of free memory in the linear address space, in pages.
dwSwapFilePages	Specifies the number of pages in the system swap file.
wPageSize	Specifies the system page size, in bytes.

See Also

[MemManInfo](#)

## MENUITEMTEMPLATE (3.0)

```
typedef struct { /* mit */
    UINT mtOption;
    UINT mtID;
    char mtString[1];
} MENUITEMTEMPLATE;
```

The MENUITEMTEMPLATE structure defines a menu item.

Member	Description																
mtOption	Specifies a mask of one or more predefined menu options that specify the appearance of the menu item. The menu options follow: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MF_CHECKED</td><td>Item has a check mark next to it.</td></tr><tr><td>MF_GRAYED</td><td>Item is initially inactive and drawn with a gray effect.</td></tr><tr><td>MF_HELP</td><td>Item has a vertical separator to its left.</td></tr><tr><td>MF_MENUBARBREAK</td><td>Item is placed in a new column. The old and new columns are separated by a bar.</td></tr><tr><td>MF_MENUBREAK</td><td>Item is placed in a new column.</td></tr><tr><td>MF_OWNERDRAW</td><td>Owner of the menu is responsible for drawing all visual aspects of the menu item, including highlighted, checked and inactive states. This option is not valid for a top-level menu item.</td></tr><tr><td>MF_POPUP</td><td>Item displays a sublist of menu items when selected.</td></tr></tbody></table>	Value	Meaning	MF_CHECKED	Item has a check mark next to it.	MF_GRAYED	Item is initially inactive and drawn with a gray effect.	MF_HELP	Item has a vertical separator to its left.	MF_MENUBARBREAK	Item is placed in a new column. The old and new columns are separated by a bar.	MF_MENUBREAK	Item is placed in a new column.	MF_OWNERDRAW	Owner of the menu is responsible for drawing all visual aspects of the menu item, including highlighted, checked and inactive states. This option is not valid for a top-level menu item.	MF_POPUP	Item displays a sublist of menu items when selected.
Value	Meaning																
MF_CHECKED	Item has a check mark next to it.																
MF_GRAYED	Item is initially inactive and drawn with a gray effect.																
MF_HELP	Item has a vertical separator to its left.																
MF_MENUBARBREAK	Item is placed in a new column. The old and new columns are separated by a bar.																
MF_MENUBREAK	Item is placed in a new column.																
MF_OWNERDRAW	Owner of the menu is responsible for drawing all visual aspects of the menu item, including highlighted, checked and inactive states. This option is not valid for a top-level menu item.																
MF_POPUP	Item displays a sublist of menu items when selected.																
mtID	Specifies an identification code for a non-pop-up menu item. The MENUITEMTEMPLATE structure for a pop-up menu item does not contain the mtID member.																
mtString	Specifies a null-terminated string that contains the name of the menu item.																

See Also

[LoadMenuIndirect](#), [MENUITEMTEMPLATEHEADER](#)

## MENUITEMTEMPLATEHEADER (3.0)

```
typedef struct { /* with */
    UINT    versionNumber;
    UINT    offset;
} MENUITEMTEMPLATEHEADER;
```

A complete menu template consists of a header and one or more menu-item lists.

Member	Description
versionNumber	Specifies the version number. This member should be zero.
offset	Specifies the offset from the end of the header, in bytes, where the menu-item list begins.

### Comments

One or more MENUITEMTEMPLATE structures are combined to form the menu-item list.

### See Also

MENUITEMTEMPLATE

## METAFILEPICT (2.x)

```
typedef struct tagMETAFILEPICT {    /* mfp */
    int  mm;
    int  xExt;
    int  yExt;
    HMETAFILE hMF;
} METAFILEPICT;
```

The METAFILEPICT structure defines the metafile picture format used for exchanging metafile data through the clipboard.

Member	Description
mm	Specifies the mapping mode in which the picture is drawn.
xExt	Specifies the size of the metafile picture for all modes except the MM_ISOTROPIC and MM_ANISOTROPIC modes. The x-extent specifies the width of the rectangle within which the picture is drawn. The coordinates are in units that correspond to the mapping mode.
yExt	Specifies the size of the metafile picture for all modes except the MM_ISOTROPIC and MM_ANISOTROPIC modes. The y-extent specifies the height of the rectangle within which the picture is drawn. The coordinates are in units that correspond to the mapping mode.  For MM_ISOTROPIC and MM_ANISOTROPIC modes, which can be scaled, the xExt and yExt members contain an optional suggested size in MM_HIMETRIC units. For MM_ANISOTROPIC pictures, xExt and yExt can be zero when no suggested size is supplied. For MM_ISOTROPIC pictures, an aspect ratio must be supplied even when no suggested size is given. (If a suggested size is given, the aspect ratio is implied by the size.) To give an aspect ratio without implying a suggested size, set xExt and yExt to negative values whose ratio is the appropriate aspect ratio. The magnitude of the negative xExt and yExt values will be ignored; only the ratio will be used.
hMF	Identifies a memory metafile.

See Also

[SetClipboardData](#)

## METAHEADER (3.1)

```
typedef struct tagMETAHEADER { /* mh */
    UINT mtType;
    UINT mtHeaderSize;
    UINT mtVersion;
    DWORD mtSize;
    UINT mtNoObjects;
    DWORD mtMaxRecord;
    UINT mtNoParameters;
} METAHEADER;
```

The METAHEADER structure contains information about a metafile.

Member	Description						
mtType	Specifies whether the metafile is in memory or recorded in a disk file. This member can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>1</td><td>Metafile is in memory.</td></tr><tr><td>2</td><td>Metafile is in a disk file.</td></tr></tbody></table>	Value	Meaning	1	Metafile is in memory.	2	Metafile is in a disk file.
Value	Meaning						
1	Metafile is in memory.						
2	Metafile is in a disk file.						
mtHeaderSize	Specifies the size, in words, of the metafile header.						
mtVersion	Specifies the Windows version number. The version number for metafiles that support device-independent bitmaps (DIBs) is 0x0300. Otherwise, the version number is 0x0100.						
mtSize	Specifies the size, in words, of the file.						
mtNoObjects	Specifies the maximum number of objects that exist in the metafile at the same time.						
mtMaxRecord	Specifies the size, in words, of the largest record in the metafile.						
mtNoParameters	Reserved.						

See Also  
[METARECORD](#)

## METARECORD (3.1)

```
typedef struct tagMETARECORD { /* mr */
    DWORD rdSize;
    UINT rdFunction;
    UINT rdParm[1];
} METARECORD;
```

The METARECORD structure contains a metafile record.

Member	Description
rdSize	Specifies the size, in words, of the record.
rdFunction	Specifies the function number.
rdParm	Specifies an array of words containing the function parameters, in the reverse order in which they are passed to the function.

See Also  
METAHEADER



## MINMAXINFO (3.1)

```
typedef struct tagMINMAXINFO { /* mmi */
    POINT ptReserved;
    POINT ptMaxSize;
    POINT ptMaxPosition;
    POINT ptMinTrackSize;
    POINT ptMaxTrackSize;
} MINMAXINFO;
```

The MINMAXINFO structure contains information about a window's maximized size and position and its minimum and maximum tracking size.

Member	Description
ptReserved	Reserved for internal use.
ptMaxSize	Specifies the maximized width (point.x) and the maximized height (point.y) of the window.
ptMaxPosition	Specifies the position of the left side of the maximized window (point.x) and the position of the top of the maximized window (point.y).
ptMinTrackSize	Specifies the minimum tracking width (point.x) and the minimum tracking height (point.y) of the window.
ptMaxTrackSize	Specifies the maximum tracking width (point.x) and the maximum tracking height (point.y) of the window.

See Also

POINT, WM\_GETMINMAXINFO

## MODULEENTRY (3.1)

```
#include <toolhelp.h>

typedef struct tagMODULEENTRY { /* me */
    DWORD    dwSize;
    char     szModule[MAX_MODULE_NAME + 1];
    HMODULE  hModule;
    WORD     wcUsage;
    char     szExePath[MAX_PATH + 1];
    WORD     wNext;
} MODULEENTRY;
```

The MODULEENTRY structure contains information about one module in the module list.

Member	Description
dwSize	Specifies the size of the MODULEENTRY structure, in bytes.
szModule	Specifies the null-terminated string that contains the module name.
hModule	Identifies the module handle.
wcUsage	Specifies the reference count of the module. This is the same number returned by the <u>GetModuleUsage</u> function.
szExePath	Specifies the null-terminated string that contains the fully-qualified executable path for the module.
wNext	Specifies the next module in the module list. This member is reserved for internal use by Windows.

See Also

ModuleFindHandle, ModuleFindName, ModuleFirst, ModuleNext

## MONCBSTRUCT (3.1)

```
#include <ddeml.h>

typedef struct tagMONCBSTRUCT { /* mcbst */
    UINTcb;
    WORDwReserved;
    DWORD dwTime;
    HANDLE hTask;
    DWORD dwRet;
    UINTwType;
    UINTwFmt;
    HCONV hConv;
    HSZ hsz1;
    HSZ hsz2;
    HDDEDATA hData;
    DWORD dwData1;
    DWORD dwData2;
} MONCBSTRUCT;
```

The MONCBSTRUCT structure contains information about the current dynamic data exchange (DDE) transaction. A DDE debugging application can use this structure when monitoring transactions that the system passes to the DDE callback functions of other applications.

Member	Description
cb	Specifies the length, in bytes, of the structure.
wReserved	Reserved.
dwTime	Specifies the Windows time at which the transaction occurred. Windows time is the number of milliseconds that have elapsed since the system was started.
hTask	Identifies the task (application instance) containing the DDE callback function that received the transaction.
dwRet	Specifies the value returned by the DDE callback function that processed the transaction.
wType	Specifies the transaction type.
wFmt	Specifies the format of the data (if any) exchanged during the transaction.
hConv	Identifies the conversation in which the transaction took place.
hsz1	Identifies a string.
hsz2	Identifies a string.
hData	Identifies the data (if any) exchanged during the transaction.
dwData1	Specifies additional data.
dwData2	Specifies additional data.

See Also

[MONERRSTRUCT](#), [MONHSZSTRUCT](#), [MONLINKSTRUCT](#), [MONMSGSTRUCT](#), [XTYP\\_MONITOR](#)

## MONCONVSTRUCT (3.1)

```
#include <ddeml.h>

typedef struct tagMONCONVSTRUCT { /* mcvst */
    UINT      cb;
    BOOL      fConnect;
    DWORD     dwTime;
    HANDLE    hTask;
    HSZhszSvc;
    HSZhszTopic;
    HCONV     hConvClient;
    HCONV     hConvServer;
} MONCONVSTRUCT;
```

The MONCONVSTRUCT structure contains information about a conversation. A dynamic data exchange (DDE) monitoring application can use this structure to obtain information about an advise loop that has been established or terminated.

Member	Description
cb	Specifies the length, in bytes, of the structure.
fConnect	Indicates whether the conversation is currently established. A value of TRUE indicates the conversation is established; FALSE indicates it is not.
dwTime	Specifies the Windows time at which the conversation was established or terminated. Windows time is the number of milliseconds that have elapsed since the system was started.
hTask	Identifies a task (application instance) that is a partner in the conversation.
hszSvc	Identifies the service name on which the conversation is established.
hszTopic	Identifies the topic name on which the conversation is established.
hConvClient	Identifies the client conversation.
hConvServer	Identifies the server conversation.

See Also

MONCBSTRUCT, MONERRSTRUCT, MONHSZSTRUCT, MONLINKSTRUCT, MONMSGSTRUCT, XTYP\_MONITOR

## MONERRSTRUCT (3.1)

```
#include <ddeml.h>

typedef struct tagMONERRSTRUCT { /* mest */
    UINT    cb;
    UINT    wLastError;
    DWORD   dwTime;
    HANDLE  hTask;
} MONERRSTRUCT;
```

The MONERRSTRUCT structure contains information about the current dynamic data exchange (DDE) error. A DDE monitoring application can use this structure to monitor errors returned by DDE Management Library functions.

Member	Description
cb	Specifies the length, in bytes, of the structure.
wLastError	Specifies the current error.
dwTime	Specifies the Windows time at which the error occurred. Windows time is the number of milliseconds that have elapsed since the system was started.
hTask	Identifies the task (application instance) that called the DDE function that caused the error.

See Also

MONCBSTRUCT, MONCONVSTRUCT, MONHSZSTRUCT, MONLINKSTRUCT, MONMSGSTRUCT, XTYP\_MONITOR

## MONHSZSTRUCT (3.1)

```
#include <ddeml.h>

typedef struct tagMONHSZSTRUCT { /* mhst */
    UINT    cb;
    BOOL    fsAction;
    DWORD   dwTime;
    HSZ     hsz;
    HANDLE  hTask;
    WORD    wReserved;
    char    str[1];
} MONHSZSTRUCT;
```

The MONHSZSTRUCT structure contains information about a dynamic data exchange (DDE) string handle. A DDE monitoring application can use this structure when monitoring the activity of the string-manager component of the DDE Management Library (DDEML).

Member	Description										
cb	Specifies the length, in bytes, of the structure.										
fsAction	Specifies the action being performed on the string handle identified by the hsz member. <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MH_CLEANUP</td><td>An application is freeing its DDE resources, causing the system to delete string handles that the application had created. (The application called the <u>DdeUninitialize</u> function.)</td></tr><tr><td>MH_CREATE</td><td>An application is creating a string handle. (The application called the <u>DdeCreateStringHandle</u> function.)</td></tr><tr><td>MH_DELETE</td><td>An application is deleting a string handle. (The application called the <u>DdeFreeStringHandle</u> function.)</td></tr><tr><td>MH_KEEP</td><td>An application is increasing the use count of a string handle. (The application called the <u>DdeKeepStringHandle</u> function.)</td></tr></tbody></table>	Value	Meaning	MH_CLEANUP	An application is freeing its DDE resources, causing the system to delete string handles that the application had created. (The application called the <u>DdeUninitialize</u> function.)	MH_CREATE	An application is creating a string handle. (The application called the <u>DdeCreateStringHandle</u> function.)	MH_DELETE	An application is deleting a string handle. (The application called the <u>DdeFreeStringHandle</u> function.)	MH_KEEP	An application is increasing the use count of a string handle. (The application called the <u>DdeKeepStringHandle</u> function.)
Value	Meaning										
MH_CLEANUP	An application is freeing its DDE resources, causing the system to delete string handles that the application had created. (The application called the <u>DdeUninitialize</u> function.)										
MH_CREATE	An application is creating a string handle. (The application called the <u>DdeCreateStringHandle</u> function.)										
MH_DELETE	An application is deleting a string handle. (The application called the <u>DdeFreeStringHandle</u> function.)										
MH_KEEP	An application is increasing the use count of a string handle. (The application called the <u>DdeKeepStringHandle</u> function.)										
dwTime	Specifies the Windows time at which the action specified by the fsAction member takes place. Windows time is the number of milliseconds that have elapsed since the system was booted.										
hsz	Identifies the string.										
hTask	Identifies the task (application instance) performing the action on the string handle.										
wReserved	Reserved.										
str	Points to the string identified by the hsz member.										

See Also

MONCBSTRUCT, MONCONVSTRUCT, MONERRSTRUCT, MONLINKSTRUCT, MONMSGSTRUCT, DdeCreateStringHandle, DdeFreeStringHandle, DdeKeepStringHandle, DdeUninitialize

## MONLINKSTRUCT (3.1)

```
#include <ddeml.h>

typedef struct tagMONLINKSTRUCT { /* mlst */
    UINT      cb;
    DWORD     dwTime;
    HANDLE    hTask;
    BOOL      fEstablished;
    BOOL      fNoData;
    HSZhszSvc;
    HSZhszTopic;
    HSZhszItem;
    UINT      wFmt;
    BOOL      fServer;
    HCONV     hConvServer;
    HCONV     hConvClient;
} MONLINKSTRUCT;
```

The MONLINKSTRUCT structure contains information about a dynamic data exchange (DDE) advise loop. A DDE monitoring application can use this structure to obtain information about an advise loop that has started or ended.

Member	Description
cb	Specifies the length, in bytes, of the structure.
dwTime	Specifies the Windows time at which the advise loop was started or ended. Windows time is the number of milliseconds that have elapsed since the system was started.
hTask	Identifies a task (application instance) that is a partner in the advise loop.
fEstablished	Indicates whether an advise loop was successfully established. A value of TRUE indicates an advise loop was established; FALSE indicates an advise loop was not established.
fNoData	Indicates whether the XTYPF_NODATA flag was set for the advise loop. A value of TRUE indicates the flag is set; FALSE indicates the flag was not set.
hszSvc	Identifies the service name of the server in the advise loop.
hszTopic	Identifies the topic name on which the advise loop is established.
hszItem	Identifies the item name that is the subject of the advise loop.
wFmt	Specifies the format of the data exchanged (if any) during the advise loop.
fServer	Indicates whether the link notification came from the server. If the notification came from the server, this value is TRUE. Otherwise, it is FALSE.
hConvServer	Identifies the server conversation.
hConvClient	Identifies the client conversation.

See Also

[MONCBSTRUCT](#), [MONERRSTRUCT](#), [MONHSZSTRUCT](#), [MONMSGSTRUCT](#), [XTYP\\_MONITOR](#)

## MONMSGSTRUCT (3.1)

```
#include <ddeml.h>

typedef struct tagMONMSGSTRUCT { /* mmst */
    UINT    cb;
    HWND    hwndTo;
    DWORD   dwTime;
    HANDLE  hTask;
    UINT    wParam;
    WPARAM  wParam;
    LPARAM  lParam;
} MONMSGSTRUCT;
```

The MONMSGSTRUCT structure contains information about a dynamic data exchange (DDE) message. A DDE monitoring application can use this structure to obtain information about a DDE message that was sent or posted.

Member	Description
cb	Specifies the length, in bytes, of the structure.
hwndTo	Identifies the window that receives the DDE message.
dwTime	Specifies the Windows time at which the message was sent or posted. Windows time is the number of milliseconds that have elapsed since the system was started.
hTask	Identifies the task (application instance) containing the window that receives the DDE message.
wMsg	Specifies the identifier of the DDE message.
wParam	Specifies the wParam parameter of the DDE message.
lParam	Specifies the lParam parameter of the DDE message.

See Also

[MONCBSTRUCT](#), [MONCONVSTRUCT](#), [MONERRSTRUCT](#), [MONHSZSTRUCT](#),  
[MONLINKSTRUCT](#), [XTYP\\_MONITOR](#)



## MOUSEHOOKSTRUCT (3.1)

```
typedef struct tagMOUSEHOOKSTRUCT { /* ms */
    POINT    pt;
    HWND     hwnd;
    UINT     wHitTestCode;
    DWORD    dwExtraInfo;
} MOUSEHOOKSTRUCT;
```

The MOUSEHOOKSTRUCT structure contains information about a mouse event.

Member	Description
pt	Specifies a <u>POINT</u> structure that contains the x- and y-coordinates of the mouse cursor, in screen coordinates.
hwnd	Identifies the window that will receive the mouse message that corresponds to the mouse event.
wHitTestCode	Specifies the hit-test code.
dwExtraInfo	Specifies extra information associated with the mouse event. An application can set this extra information by calling the <u>hardware_event</u> function and retrieve it by calling the <u>GetMessageExtraInfo</u> function.

See Also

hardware\_event, GetMessageExtraInfo, SetWindowsHook

## MSG (2.x)

```
typedef struct tagMSG { /* msg */
    HWND    hwnd;
    UINT    message;
    WPARAM  wParam;
    LPARAM  lParam;
    DWORD   time;
    POINT   pt;
} MSG;
```

The MSG structure contains information from the Windows application queue.

Member	Description
hwnd	Identifies the window that receives the message.
message	Specifies the message number.
wParam	Specifies additional information about the message. The exact meaning depends on the message value.
lParam	Specifies additional information about the message. The exact meaning depends on the message value.
time	Specifies the time at which the message was posted.
pt	Specifies the position of the cursor, in screen coordinates, when the message was posted.

See Also

[EVENTMSG](#), [GetMessage](#)

## MULTIKEYHELP (3.0)

```
typedef struct tagMULTIKEYHELP {      /* mkh */
    UINT    mkSize;
    BYTE    mkKeylist;
    BYTE    szKeyphrase[1];
} MULTIKEYHELP;
```

The MULTIKEYHELP structure specifies a keyword table and an associated keyword to be used by the Windows Help application.

Member	Description
mkSize	Specifies the length, in bytes, of the MULTIKEYHELP structure.
mkKeylist	Contains a single character that identifies the keyword table to be searched.
szKeyphrase	Contains a null-terminated text string that specifies the keyword to be located in the keyword table.

See Also

[WinHelp](#)

## NCCALCSIZE\_PARAMS (3.1)

```
typedef struct tagNCCALCSIZE_PARAMS {  
    RECT rgrc[3];  
    WINDOWPOS FAR* lppos;  
} NCCALCSIZE_PARAMS;
```

The `NCCALCSIZE_PARAMS` structure contains information that an application can use while processing the `WM_NCCALCSIZE` message to calculate the size, position, and valid contents of the client area of a window.

Member	Description
<code>rgrc</code>	Specifies an array of rectangles. The first contains the new coordinates of a window that has been moved or resized. The second contains the coordinates of the window before it was moved or resized. The third contains the coordinates of the client area of a window before it was moved or resized. If the window is a child window, the coordinates are relative to the client area of the parent window. If the window is a top-level window, the coordinates are relative to the screen.
<code>lppos</code>	Points to a <code>WINDOWPOS</code> structure that contains the size and position values specified in the operation that caused the window to be moved or resized.

See Also

[MoveWindow](#), [SetWindowPos](#), [RECT](#), [WINDOWPOS](#), [WM\\_NCCALCSIZE](#)

## NEWCPLINFO (3.1)

```
#include <cpl.h>

typedef struct tagNEWCPLINFO { /* ncpli */
    DWORD    dwSize;
    DWORD    dwFlags;
    DWORD    dwHelpContext;
    LONG     lData;
    HICON    hIcon;
    char     szName[32];
    char     szInfo[64];
    char     szHelpFile[128];
} NEWCPLINFO;
```

The NEWCPLINFO structure contains resource information and a user-defined value for a Control Panel application.

Member	Description
dwSize	Specifies the length of the structure, in bytes.
dwFlags	Specifies Control Panel flags.
dwHelpContext	Specifies the context number for the topic in the help project (.HPJ) file that displays when the user selects help for the application.
lData	Specifies data defined by the application.
hIcon	Identifies an icon resource for the application icon. This icon is displayed in the Control Panel window.
szName	Specifies a null-terminated string that contains the application name. The name is the short string displayed below the application icon in the Control Panel window. The name is also displayed in the Settings menu of Control Panel.
szInfo	Specifies a null-terminated string containing the application description. The description displayed at the bottom of the Control Panel window when the application icon is selected.
szHelpFile	Specifies a null-terminated string that contains the path of the help file, if any, for the application.

## NEWTEXTMETRIC (3.1)

```
typedef struct tagNEWTEXTMETRIC { /* ntm */
    int    tmHeight;
    int    tmAscent;
    int    tmDescent;
    int    tmInternalLeading;
    int    tmExternalLeading;
    int    tmAveCharWidth;
    int    tmMaxCharWidth;
    int    tmWeight;
    BYTE   tmItalic;
    BYTE   tmUnderlined;
    BYTE   tmStruckOut;
    BYTE   tmFirstChar;
    BYTE   tmLastChar;
    BYTE   tmDefaultChar;
    BYTE   tmBreakChar;
    BYTE   tmPitchAndFamily;
    BYTE   tmCharSet;
    int    tmOverhang;
    int    tmDigitizedAspectX;
    int    tmDigitizedAspectY;
    DWORD  ntmFlags;
    UINT   ntmSizeEM;
    UINT   ntmCellHeight;
    UINT   ntmAvgWidth;
} NEWTEXTMETRIC;
```

The NEWTEXTMETRIC structure contains basic information about a physical font. The last four members of the NEWTEXTMETRIC structure are not included in the TEXTMETRIC structure; in all other respects, the structures are identical. The additional members are used for information about TrueType fonts.

Member	Description
tmHeight	Specifies the height of character cells. (The height is the sum of the tmAscent and tmDescent members.)
tmAscent	Specifies the ascent of character cells. (The ascent is the space between the base line and the top of the character cell.)
tmDescent	Specifies the descent of character cells. (The descent is the space between the bottom of the character cell and the base line.)
tmInternalLeading	Specifies the difference between the point size of a font and the physical size of the font. For TrueType fonts, this value is equal to tmHeight minus (s * ntmSizeEM), where s is the scaling factor for the TrueType font. For bitmap fonts, this value is used to determine the point size of a font; when an application specifies a negative value in the lfHeight member of the LOGFONT structure, the application is requesting a font whose height equals tmHeight minus tmInternalLeading.
tmExternalLeading	Specifies the amount of extra leading (space) that the application adds between rows. Since this area is outside the character cell, it contains no marks and will not be altered by text output calls in either opaque or transparent mode. The font designer sometimes sets this member to zero.
tmAveCharWidth	Specifies the average width of characters in the font. For ANSI_CHARSET fonts, this is a weighted average of the characters "a" through "z" and the space character. For other character sets, this value is an unweighted average of all characters in the font.
tmMaxCharWidth	Specifies the "B" spacing of the widest character in the font. For more information about "B" spacing, see the description of the ABC structure.
tmWeight	Specifies the weight of the font. This member can be one of the following values:
Constant	Value

	FW_DONTCARE	0
	FW_THIN	100
	FW_EXTRALIGHT	200
	FW_ULTRALIGHT	200
	FW_LIGHT	300
	FW_NORMAL	400
	FW_REGULAR	400
	FW_MEDIUM	500
	FW_SEMIBOLD	600
	FW_DEMIBOLD	600
	FW_BOLD	700
	FW_EXTRABOLD	800
	FW_ULTRABOLD	800
	FW_BLACK	900
	FW_HEAVY	900
tmItalic	Specifies an italic font if it is nonzero.	
tmUnderlined	Specifies an underlined font if it is nonzero.	
tmStruckOut	Specifies a "struckout" font if it is nonzero.	
tmFirstChar	Specifies the value of the first character defined in the font.	
tmLastChar	Specifies the value of the last character defined in the font.	
tmDefaultChar	Specifies the value of the character that will be substituted for characters not in the font.	
tmBreakChar	Specifies the value of the character that will be used to define word breaks for text justification.	
tmPitchAndFamily	Specifies the pitch and family of the selected font. The four low-order bits identify the type of font, as follows:	
	Value	Meaning
	TMPF_FIXED_PITCH	Designates a fixed-pitch font.
	TMPF_VECTOR	Designates a vector or TrueType font.
	TMPF_TRUETYPE	Designates a TrueType font.
	TMPF_DEVICE	Designates a device font.
	Some fonts are identified by several of these bits--for example, the TMPF_FIXED_PITCH, TMPF_VECTOR, and TMPF_TRUETYPE bits would be set for the monospace TrueType font, Courier New®. The TMPF_DEVICE bit could be set for a TrueType font as well, because this bit is set both for downloaded and device-resident fonts.	
	When the TMPF_TRUETYPE bit is set, the font is usable on all output devices. For example, if a TrueType font existed on a printer but could not be used on the display, the TMPF_TRUETYPE bit would not be set for that font.	
	The four high-order bits specify the font family. The tmPitchAndFamily member can be combined with the hexadecimal value 0xF0 by using the bitwise AND operator and can then be compared with the font family names for an identical match. The following font families are defined:	
	Value	Meaning
	FF_DECORATIVE	Novelty fonts. Old English is an example.
	FF_DONTCARE	Don't care or don't know.
	FF_MODERN	Fonts with constant stroke width, with or without serifs. Pica, Elite, and Courier New are examples.
	FF_ROMAN	Fonts with variable stroke width and with serifs. Times New Roman and New Century Schoolbook are examples.
	FF_SCRIPT	Fonts designed to look like handwriting. Script and Cursive are examples.

	FF_SWISS	Fonts with variable stroke width and without serifs. <u>MS Sans Serif</u> is an example.
tmCharSet	Specifies the character set of the font. The following values are defined:	
	Constant	Value
	ANSI_CHARSET	0
	DEFAULT_CHARSET	1
	SYMBOL_CHARSET	2
	SHIFTJIS_CHARSET	128
	OEM_CHARSET	255
tmOverhang	Specifies the extra width that is added to some synthesized fonts. When synthesizing some attributes, such as bold or italic, graphics-device interface (GDI) or a device adds width to a string on both a per-character and per-string basis. For example, GDI makes a string bold by expanding the intracharacter spacing and overstriking by an offset value and italicizes a font by skewing the string. In either case, the string is wider after the attribute is synthesized. For bold strings, the overhang is the distance by which the overstrike is offset. For italic strings, the overhang is the amount the top of the font is skewed past the bottom of the font.	
	The tmOverhang member is zero for many italic and bold TrueType fonts because many TrueType fonts include italic and bold faces that are not synthesized. For example, the overhang for Courier New®Italic is zero.	
	An application that uses raster fonts can use the overhang value to determine the spacing between words that have different attributes.	
tmDigitizedAspectX	Specifies the horizontal aspect of the device for which the font was designed.	
tmDigitizedAspectY	Specifies the vertical aspect of the device for which the font was designed. The ratio of the tmDigitizedAspectX and tmDigitizedAspectY members is the aspect ratio of the device for which the font was designed.	
ntmFlags	Specifies some elements of the font style. This member can be one or more of the following values:	
	NTM_REGULAR	
	NTM_BOLD	
	NTM_ITALIC	
	The NTM_BOLD and NTM_ITALIC flags could be combined with the OR operator to specify a bold italic font.	
ntmSizeEM	Specifies the size of the em square for the font, in the units for which the font was designed (notional units).	
ntmCellHeight	Specifies the height of the font, in the units for which the font was designed (notional units). This value should be compared against the value of the ntmSizeEM member.	
ntmAvgWidth	Specifies the average width of characters in the font, in the units for which the font was designed (notional units). This value should be compared against the value of the ntmSizeEM member.	

**Comments**

The sizes in the NEWTEXTMETRIC structure are typically given in logical units; that is, they depend on the current mapping mode of the display context.

See Also

[EnumFontFamilies](#), [EnumFonts](#), [GetDeviceCaps](#), [GetTextMetrics](#), [TEXTMETRIC](#)



## NFYLOADSEG (3.1)

```
#include <toolhelp.h>

typedef struct tagNFYLOADSEG { /* nfyls */
    DWORD    dwSize;
    WORD     wSelector;
    WORD     wSegNum;
    WORD     wType;
    WORD     wcInstance;
    LPCSTR   lpstrModuleName;
} NFYLOADSEG;
```

The NFYLOADSEG structure contains information about the segment being loaded when the kernel sends a load-segment notification.

Member	Description						
dwSize	Specifies the size of the NFYLOADSEG structure, in bytes.						
wSelector	Contains the selector of the segment being loaded.						
wSegNum	Contains the executable-file segment number.						
wType	Indicates the type of information in the segment. Only the low bit of wType is used. This type can be one of the following values:						
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>The segment contains code.</td></tr><tr><td>1</td><td>The segment contains data.</td></tr></tbody></table>	Value	Meaning	0	The segment contains code.	1	The segment contains data.
Value	Meaning						
0	The segment contains code.						
1	The segment contains data.						
wcInstance	Specifies the number of instances that share this segment. This value is valid only for data segments.						
lpstrModuleName	Points to a null-terminated string containing the name of the module that owns the segment being loaded.						

See Also  
[NotifyRegister](#)

## NFYLOGERROR (3.1)

```
#include <toolhelp.h>
```

```
typedef struct tagNFYLOGERROR { /* nfyle */  
    DWORD dwSize;  
    UINT wErrCode;  
    void FAR* lpInfo;  
} NFYLOGERROR;
```

The NFYLOGERROR structure contains information about a validation error that caused the kernel to send an NFY\_LOGERROR notification.

Member	Description
dwSize	Specifies the size of the NFYLOGERROR structure, in bytes.
wErrCode	Identifies the error value that caused the notification to be sent.
lpInfo	Points to additional information, dependent on the error value.

See Also

[NotifyRegister](#)

## NFYLOGPARAMERROR (3.1)

```
#include <toolhelp.h>
```

```
typedef struct tagNFYLOGPARAMERROR { /* nfylpe */
    DWORD dwSize;
    UINT wErrCode;
    FARPROC lpfnErrorAddr;
    void FAR* FAR* lpBadParam;
} NFYLOGPARAMERROR;
```

The NFYLOGPARAMERROR structure contains information about a parameter-validation error that caused the kernel to send an NFY\_LOGPARAMERROR notification.

Member	Description
dwSize	Specifies the size of the NFYLOGPARAMERROR structure, in bytes.
wErrCode	Identifies the error value that caused the notification to be sent.
lpfnErrorAddr	Points to the address of the function with the invalid parameter.
lpBadParam	Points to the name of the invalid parameter.

See Also

[NotifyRegister](#)

## NFYRIP (3.1)

```
#include <toolhelp.h>

typedef struct tagNFYRIP { /* nfyr */
    DWORD dwSize;
    WORD wIP;
    WORD wCS;
    WORD wSS;
    WORD wBP;
    WORD wExitCode;
} NFYRIP;
```

The NFYRIP structure contains information about the system when a system debugging error (RIP) occurs.

Member	Description
dwSize	Specifies the size of the NFYRIP structure, in bytes.
wIP	Contains the value in the IP register at the time of the RIP.
wCS	Contains the value in the CS register at the time of the RIP.
wSS	Contains the value in the SS register at the time of the RIP.
wBP	Contains the value in the BP register at the time of the RIP.
wExitCode	Contains an exit code that describes why the RIP occurred.

### Comments

The StackTraceCSIPFirst function uses the CS:IP and SS:BP values presented in this structure. The first frame in the stack identified by these values points to the FatalExit function. The next frame points to the routine that called FatalExit, usually in USER.EXE, GDI.EXE, or either KRNL286.EXE or KRNL386.EXE.

### See Also

FatalExit, NotifyRegister, StackTraceCSIPFirst

## NFYSTARTDLL (3.1)

```
#include <toolhelp.h>

typedef struct tagNFYSTARTDLL { /* nfysd */
    DWORD    dwSize;
    HMODULE  hModule;
    WORD     wCS;
    WORD     wIP;
} NFYSTARTDLL;
```

The NFYSTARTDLL structure contains information about the dynamic-link library (DLL) being loaded when the kernel sends a load-DLL notification.

Member	Description
dwSize	Specifies the size of the NFYSTARTDLL structure, in bytes.
hModule	Identifies the library module being loaded.
wCS	Contains the value in the <u>CS</u> register at load time. This value is used with the value of the wIP member to determine the load address of the library.
wIP	Contains the value in the IP register at load time. This value is used with the wCS value to determine the load address of the library.

See Also  
[NotifyRegister](#)

## OFSTRUCT (2.x)

```
typedef struct tagOFSTRUCT {      /* of */
    BYTE  cBytes;
    BYTE  fFixedDisk;
    UINT  nErrCode;
    BYTE  reserved[4];
    BYTE  szPathName[128];
} OFSTRUCT;
```

The OFSTRUCT structure contains file information which results from opening that file.

Member	Description
cBytes	Specifies the length, in bytes, of the OFSTRUCT structure.
fFixedDisk	Specifies whether the file is on a fixed disk. The fFixedDisk member is nonzero if the file is on a fixed disk.
nErrCode	Specifies the MS-DOS error value if the <code>OpenFile</code> function returns -1 (that is, <code>OpenFile</code> fails). For a list of possible error values, see the following Comments section.
reserved	Reserved member. Four bytes reserved for future use.
szPathName	Specifies 128 bytes that contain the path of the file. This string consists of characters from the OEM character set.

### Comments

The error values that may be specified in the nErrCode parameter follow:

Value	Meaning
0x0001	Invalid function
0x0002	File not found
0x0003	Path not found
0x0004	Too many open files
0x0005	Access denied
0x0006	Invalid handle
0x0007	Arena trashed
0x0008	Not enough memory
0x0009	Invalid block
0x000A	Bad environment
0x000B	Bad format
0x000C	Invalid access
0x000D	Invalid data
0x000F	Invalid drive
0x0010	Current directory
0x0011	Not same device
0x0012	No more files
0x0013	Write protect error
0x0014	Bad unit
0x0015	Not ready
0x0016	Bad command
0x0017	CRC error
0x0018	Bad length
0x0019	Seek error
0x001A	Not MS-DOS disk
0x001B	Sector not found
0x001C	Out of paper
0x001D	Write fault
0x001E	Read fault
0x001F	General failure

0x0020	Sharing violation
0x0021	Lock violation
0x0022	Wrong disk
0x0023	File control block unavailable
0x0024	Sharing buffer exceeded
0x0032	Not supported
0x0033	Remote not listed
0x0034	Duplicate name
0x0035	Bad netpath
0x0036	Network busy
0x0037	Device does not exist
0x0038	Too many commands
0x0039	Adaptor hardware error
0x003A	Bad network response
0x003B	Unexpected network error
0x003C	Bad remote adaptor
0x003D	Print queue full
0x003E	No spool space
0x003F	Print canceled
0x0040	Netname deleted
0x0041	Network access denied
0x0042	Bad device type
0x0043	Bad network name
0x0044	Too many names
0x0045	Too many sessions
0x0046	Sharing paused
0x0047	Request not accepted
0x0048	Redirection paused
0x0050	File exists
0x0051	Duplicate file control block
0x0052	Cannot make
0x0053	Interrupt 24 failure
0x0054	Out of structures
0x0055	Already assigned
0x0056	Invalid password
0x0057	Invalid parameter
0x0058	Net write fault

See Also  
[OpenFile](#)

## OLECLIENT (3.1)

```
#include <ole.h>

typedef struct _OLECLIENT { /* oc */
    LPOLECLIENTVTBL lpvtbl;
    . /* any client-supplied state information */
    .
} OLECLIENT;
```

The OLECLIENT structure points to an OLECLIENTVTBL structure and can store state information for use by the client application.

Member	Description
lpvtbl	Points to a table of function pointers for the client.

### Comments

Servers and object handlers should not attempt to use any state information supplied in the OLECLIENT structure. The use and meaning of this information is entirely dependent on the client application. Because a pointer to this structure is supplied as a parameter to the client's callback function, this is the preferred method for the client application to store private object-state information.

### See Also

OLECLIENTVTBL



## OLECLIENTVTBL (3.1)

```
#include <ole.h>
```

```
typedef struct _OLECLIENTVTBL { /* ocv */  
    int (CALLBACK* CallBack)(LPOLECLIENT, OLE_NOTIFICATION,  
LPOLEOBJECT);  
} OLECLIENTVTBL;
```

The OLECLIENTVTBL structure contains a pointer to a callback function for the client application.

Member	Description
<u>ClientCallback</u>	Points to a callback function for the client application.

### Comments

The address passed as the CallBack member must be created by using the MakeProcInstance function.

### Function

### See Also

OLECLIENT

## OLECLIENT (1.x)

INT ClientCallback(lpclient, notification, lpobject)  
LPOLECLIENT lpclient;  
OLE\_NOTIFICATION notification;  
LPOLEOBJECT lpobject;

The ClientCallback function must use the Pascal calling convention and must be declared FAR.

Parameter	Description																
lpclient	Points to the client structure associated with the object. The library retrieves this pointer from its object structure when a notification occurs, uses it to locate the callback function, and passes the pointer to the client structure for the client application's use.																
notification	Specifies the reason for the notification. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>OLE_CHANGED</td><td>The linked object has changed. (This notification is not sent for embedded objects.) A typical action to take with this notification is either to redraw or to save the object.</td></tr><tr><td>OLE_CLOSED</td><td>The object has been closed in its server. When the client receives this notification, it should not call any function that causes an asynchronous operation until it regains control of program execution.</td></tr><tr><td>OLE_QUERY_PAINT</td><td>A lengthy drawing operation is occurring. This notification allows the drawing to be interrupted.</td></tr><tr><td>OLE_QUERY_RETRY</td><td>The server has responded to a request by indicating that it is busy. This notification requests the client to determine whether the library should continue to make the request. If the callback function returns FALSE, the transaction with the server is discontinued.</td></tr><tr><td>OLE_RELEASE</td><td>The object has been released because an asynchronous operation has finished. The client should not quit until all objects have been released. The client application can call the <u>OleQueryReleaseError</u> function to determine whether the operation succeeded. It can also call the <u>OleQueryReleaseMethod</u> function, if necessary, to verify that that operation has ended.</td></tr><tr><td>OLE_RENAMED</td><td>The linked object has been renamed in its server. This notification is for information only, because the library automatically updates its link information.</td></tr><tr><td>OLE_SAVED</td><td>The linked object has been saved in its server. The client receives this notification when the server calls the <u>OleSavedServerDoc</u> function in response to the user choosing the Update command in the server's File menu.</td></tr></tbody></table> <p>When the client receives the OLE_CLOSED notification, it typically stores the condition and returns to the client library, taking action only when the client library returns control of program execution to the client application. If the client application must take action before regaining control, it should not call any functions that could result in an asynchronous operation.</p>	Value	Meaning	OLE_CHANGED	The linked object has changed. (This notification is not sent for embedded objects.) A typical action to take with this notification is either to redraw or to save the object.	OLE_CLOSED	The object has been closed in its server. When the client receives this notification, it should not call any function that causes an asynchronous operation until it regains control of program execution.	OLE_QUERY_PAINT	A lengthy drawing operation is occurring. This notification allows the drawing to be interrupted.	OLE_QUERY_RETRY	The server has responded to a request by indicating that it is busy. This notification requests the client to determine whether the library should continue to make the request. If the callback function returns FALSE, the transaction with the server is discontinued.	OLE_RELEASE	The object has been released because an asynchronous operation has finished. The client should not quit until all objects have been released. The client application can call the <u>OleQueryReleaseError</u> function to determine whether the operation succeeded. It can also call the <u>OleQueryReleaseMethod</u> function, if necessary, to verify that that operation has ended.	OLE_RENAMED	The linked object has been renamed in its server. This notification is for information only, because the library automatically updates its link information.	OLE_SAVED	The linked object has been saved in its server. The client receives this notification when the server calls the <u>OleSavedServerDoc</u> function in response to the user choosing the Update command in the server's File menu.
Value	Meaning																
OLE_CHANGED	The linked object has changed. (This notification is not sent for embedded objects.) A typical action to take with this notification is either to redraw or to save the object.																
OLE_CLOSED	The object has been closed in its server. When the client receives this notification, it should not call any function that causes an asynchronous operation until it regains control of program execution.																
OLE_QUERY_PAINT	A lengthy drawing operation is occurring. This notification allows the drawing to be interrupted.																
OLE_QUERY_RETRY	The server has responded to a request by indicating that it is busy. This notification requests the client to determine whether the library should continue to make the request. If the callback function returns FALSE, the transaction with the server is discontinued.																
OLE_RELEASE	The object has been released because an asynchronous operation has finished. The client should not quit until all objects have been released. The client application can call the <u>OleQueryReleaseError</u> function to determine whether the operation succeeded. It can also call the <u>OleQueryReleaseMethod</u> function, if necessary, to verify that that operation has ended.																
OLE_RENAMED	The linked object has been renamed in its server. This notification is for information only, because the library automatically updates its link information.																
OLE_SAVED	The linked object has been saved in its server. The client receives this notification when the server calls the <u>OleSavedServerDoc</u> function in response to the user choosing the Update command in the server's File menu.																
lpobject	Points to the object that caused the notification to be sent. Applications that use the same client structure for more than one object use the lpobject parameter to distinguish between notifications.																

### Returns

When the notification parameter specifies either OLE\_QUERY\_PAINT or OLE\_QUERY\_RETRY, the client should return TRUE if the library should continue, or FALSE to terminate the painting operation or discontinue the server transaction. When the notification parameter does not specify either OLE\_QUERY\_PAINT or OLE\_QUERY\_RETRY, the return value is ignored.

### Comments

The client application should act on these notifications at the next appropriate time; for example, as part of the main event loop or when closing the object. The updating of an object can be deferred until the user requests the update, if the client provides that functionality. The client may call the library from a notification callback function (the library is reentrant). The client should not attempt an asynchronous operation while certain other operations are in progress (for example, opening or deleting an object). The client also should not enter a message-dispatch loop inside the callback function. When the client application calls a function that would cause an asynchronous operation, the client library returns `OLE_WAIT_FOR_RELEASE` when the function is called, notifies the application when the operation completes by using `OLE_RELEASE`, and returns `OLE_BUSY` if the client attempts to invoke a conflicting operation while the previous one is in progress. The client can determine if an asynchronous operation is in progress by calling `OleQueryReleaseStatus`, which returns `OLE_BUSY` if the operation has not yet completed.

See Also

`OleQueryReleaseStatus`, `OLECLIENT`

## OLEOBJECT (3.1)

```
#include <ole.h>
```

```
typedef struct _OLEOBJECT { /* oo */
    LPOLEOBJECTVTBL lpvtbl;
    . /* any server-supplied state information */
    .
} OLEOBJECT;
```

The OLEOBJECT structure points to a table of function pointers for an object. This structure is initialized and maintained by servers for the server library.

Member	Description
lpvtbl	Points to a table of function pointers for the object.

See Also

[OLEOBJECTVTBL](#)

## OLEOBJECTVTBL (3.1)

```
#include <ole.h>

typedef struct _OLEOBJECTVTBL { /* oov */
    void FAR* (CALLBACK* QueryProtocol)(LPOLEOBJECT, OLE_LPCSTR);
    OLESTATUS (CALLBACK* Release)(LPOLEOBJECT);
    OLESTATUS (CALLBACK* Show)(LPOLEOBJECT, BOOL);
    OLESTATUS (CALLBACK* DoVerb)(LPOLEOBJECT, UINT, BOOL, BOOL);
    OLESTATUS (CALLBACK* GetData)(LPOLEOBJECT, OLECLIPFORMAT,
    HANDLE FAR*);
    OLESTATUS (CALLBACK* SetData)(LPOLEOBJECT, OLECLIPFORMAT, HANDLE);
    OLESTATUS (CALLBACK* SetTargetDevice)(LPOLEOBJECT, HGLOBAL);
    OLESTATUS (CALLBACK* SetBounds)(LPOLEOBJECT, OLE_CONST RECT FAR*);
    OLECLIPFORMAT (CALLBACK* EnumFormats)(LPOLEOBJECT, OLECLIPFORMAT);
    OLESTATUS (CALLBACK* SetColorScheme)(LPOLEOBJECT,
    OLE_CONST LOGPALETTE FAR*);

    /*
    * Server applications implement only the functions listed above.
    * Object handlers can use any of the functions in this structure
    * to modify default server behavior.
    */

    OLESTATUS (CALLBACK* Delete)(LPOLEOBJECT);
    OLESTATUS (CALLBACK* SetHostNames)(LPOLEOBJECT, OLE_LPCSTR,
    OLE_LPCSTR);
    OLESTATUS (CALLBACK* SaveToStream)(LPOLEOBJECT, LPOLESTREAM);
    OLESTATUS (CALLBACK* Clone)(LPOLEOBJECT, LPOLECLIENT, LHCLIENTDOC,
    OLE_LPCSTR, LPOLEOBJECT FAR*);
    OLESTATUS (CALLBACK* CopyFromLink)(LPOLEOBJECT, LPOLECLIENT,
    LHCLIENTDOC, OLE_LPCSTR, LPOLEOBJECT FAR*);
    OLESTATUS (CALLBACK* Equal)(LPOLEOBJECT, LPOLEOBJECT);
    OLESTATUS (CALLBACK* CopyToClipboard)(LPOLEOBJECT);
    OLESTATUS (CALLBACK* Draw)(LPOLEOBJECT, HDC, OLE_CONST RECT FAR*,
    OLE_CONST RECT FAR*, HDC);
    OLESTATUS (CALLBACK* Activate)(LPOLEOBJECT, UINT, BOOL, BOOL, HWND,
    OLE_CONST RECT FAR*);
    OLESTATUS (CALLBACK* Execute)(LPOLEOBJECT, HGLOBAL, UINT);
    OLESTATUS (CALLBACK* Close)(LPOLEOBJECT);
    OLESTATUS (CALLBACK* Update)(LPOLEOBJECT);
    OLESTATUS (CALLBACK* Reconnect)(LPOLEOBJECT);
    OLESTATUS (CALLBACK* ObjectConvert)(LPOLEOBJECT, OLE_LPCSTR,
    LPOLECLIENT, LHCLIENTDOC, OLE_LPCSTR, LPOLEOBJECT FAR*);
    OLESTATUS (CALLBACK* GetLinkUpdateOptions)(LPOLEOBJECT,
    OLEOPT_UPDATE FAR*);
    OLESTATUS (CALLBACK* SetLinkUpdateOptions)(LPOLEOBJECT,
    OLEOPT_UPDATE);
    OLESTATUS (CALLBACK* Rename)(LPOLEOBJECT, OLE_LPCSTR);
    OLESTATUS (CALLBACK* QueryName)(LPOLEOBJECT, LPSTR, UINT FAR*);
    OLESTATUS (CALLBACK* QueryType)(LPOLEOBJECT, LONG FAR*);
    OLESTATUS (CALLBACK* QueryBounds)(LPOLEOBJECT, RECT FAR*);
    OLESTATUS (CALLBACK* QuerySize)(LPOLEOBJECT, DWORD FAR*);
    OLESTATUS (CALLBACK* QueryOpen)(LPOLEOBJECT);
    OLESTATUS (CALLBACK* QueryOutOfDate)(LPOLEOBJECT);
    OLESTATUS (CALLBACK* QueryReleaseStatus)(LPOLEOBJECT);
    OLESTATUS (CALLBACK* QueryReleaseError)(LPOLEOBJECT);
    OLE_RELEASE_METHOD (CALLBACK* QueryReleaseMethod)(LPOLEOBJECT);
    OLESTATUS (CALLBACK* RequestData)(LPOLEOBJECT, OLECLIPFORMAT);
    OLESTATUS (CALLBACK* ObjectLong)(LPOLEOBJECT, UINT, LONG FAR*);
} OLEOBJECTVTBL;
```

The OLEOBJECTVTBL structure points to functions that manipulate an object. A server application creates this structure and an OLEOBJECT structure to give the server library access to an object.

Server applications do not need to implement functions beyond the SetColorScheme function. Object handlers can provide specialized treatment for some or all of the functions in the OLEOBJECTVTBL structure.

The following list of structure members does not document all the functions pointed to by the OLEOBJECTVTBL structure. For information about the functions not documented here, see the documentation for the corresponding function for object linking and embedding (OLE). For example, for more information about the QueryProtocol member, see the OleQueryProtocol function.

Member	Description
<u>Release</u>	Causes the server to free the resources associated with the specified <u>OLEOBJECT</u> structure.
<u>Show</u>	Causes the server to show an object.
<u>DoVerb</u>	Specifies what kind of action the server should take when a user opens an object.
<u>GetData</u>	Retrieves data from an object in a specified format.
<u>SetData</u>	Stores data in an object in a specified format.
<u>SetTargetDevice</u>	Communicates information about the client's target device for the object.
<u>SetColorScheme</u>	Sends the server application the color palette recommended by the client application.
<u>ObjectLong</u>	Allows the calling application to store data with an object. This function is typically used by object handlers.

#### Comments

The following functions in OLEOBJECTVTBL should return OLE\_BUSY when appropriate:

<u>Activate</u>	<u>SetBounds</u>
<u>Close</u>	<u>SetColorScheme</u>
<u>CopyFromLink</u>	<u>SetData</u>
<u>Delete</u>	<u>SetHostNames</u>
<u>DoVerb</u>	<u>SetLinkUpdateOptions</u>
<u>Execute</u>	<u>SetTargetDevice</u>
<u>ObjectConvert</u>	<u>Show</u>
<u>Reconnect</u>	<u>Update</u>

RequestData  
Function

See Also

OLEOBJECT

## Release (OLE 1.x)

```
OLESTATUS (FAR PASCAL *Release)(lpObject)  
LPOLEOBJECT lpObject;
```

The Release function causes the server to free the resources associated with the specified OLEOBJECT structure.

Parameter	Description
lpObject	Points to the <u>OLEOBJECT</u> structure to be released.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

The server application should not destroy data when the library calls the Release function. The library calls the Release function when no clients are connected to the object.

### Function

## Show (OLE 1.x)

```
OLESTATUS (FAR PASCAL *Show)(lpObject, fTakeFocus)
LPOLEOBJECT lpObject;
BOOL fTakeFocus;
```

The Show function causes the server to show an object, displaying its window and scrolling (if necessary) to make the object visible.

Parameter	Description
lpObject	Points to the <u>OLEOBJECT</u> structure to show.
fTakeFocus	Specifies whether the server window gets the focus. If the server window is to get the focus, this value is TRUE. Otherwise, this value is FALSE.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

The library calls the Show function when the server application should show the document to the user for editing or to request the server to scroll the document to bring the object into view.

### Function



## DoVerb (OLE 1.x)

```
OLESTATUS (FAR PASCAL *DoVerb)(lpObject, iVerb, fShow, fTakeFocus);
LPOLEOBJECT lpObject;
UINT iVerb;
BOOL fShow;
BOOL fTakeFocus;
```

The DoVerb function specifies what kind of action the server should take when a user activates an object.

Parameter	Description
lpObject	Points to the object to activate.
iVerb	Specifies the action to take. The meaning of this parameter is determined by the server application.
fShow	Specifies whether to show the server window. This value is TRUE to show the window; otherwise, it is FALSE.
fTakeFocus	Specifies whether the server window gets the focus. If the server window is to get the focus, this value is TRUE. Otherwise, it is FALSE. This parameter is relevant only if the fShow parameter is TRUE.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

All servers must support the editing of objects. If a server does not support any verbs except Edit, it should edit the object no matter what value is specified by the iVerb parameter.

### Function

## GetData (OLE 1.x)

```
OLESTATUS (FAR PASCAL *GetData)(lpObject, cfFormat, lphdata)
LPOLEOBJECT lpObject;
OLECLIPFORMAT cfFormat;
HANDLE FAR* lphdata;
```

The GetData function retrieves data from an object in a specified format. The server application should allocate memory, fill it with the data, and return the data through the lphdata parameter.

Parameter	Description
lpObject	Points to the <u>OLEOBJECT</u> structure from which data is requested.
cfFormat	Specifies the format in which the data is requested.
lphdata	Points to the handle of the allocated memory that the server application returns. The library frees the memory when it is no longer needed.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value, which may be one of the following:

```
OLE_ERROR_BLANK
OLE_ERROR_FORMAT
OLE_ERROR_OBJECT
```

Function

## SetData (OLE 1.x)

```
OLESTATUS (FAR PASCAL *SetData)(lpObject, cfFormat, hdata)
LPOLEOBJECT lpObject;
OLECLIPFORMAT cfFormat;
HANDLE hdata;
```

The SetData function stores data in an object in a specified format. This function is called (with the Native data format) when a client opens an embedded object for editing. This function is also used if the client calls the OleSetData function with some other format.

Parameter	Description
lpObject	Points to the <u>OLEOBJECT</u> structure in which data is stored.
cfFormat	Specifies the format of the data.
hdata	Identifies a place in memory from which the server application should extract the data. The server should delete this handle after it uses the data.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

The server application is responsible for the memory identified by the hdata parameter. The server must delete this data even if it returns OLE\_BUSY or if an error occurs.

### Function

## SetTargetDevice (OLE 1.x)

OLESTATUS (FAR PASCAL \*SetTargetDevice)(lpObject, hotd)  
LPOLEOBJECT lpObject;  
HGLOBAL hotd;

The SetTargetDevice function communicates information about the client's target device for the object. The server can use this information to customize output for the target device.

Parameter	Description
lpObject	Points to the <u>OLEOBJECT</u> structure for which the target device is specified.
hotd	Identifies an <u>OLETARGETDEVICE</u> structure.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

The server application is responsible for the memory identified by the hotd parameter. The server must delete this data even if it returns OLE\_BUSY or if an error occurs.

The library passes NULL for the hotd parameter to indicate that the rendering is necessary for the screen.

### See Also

OleSetTargetDevice, OLETARGETDEVICE Function

## ObjectLong (OLE 1.x)

```
OLESTATUS (FAR PASCAL *ObjectLong)(lpObject, wFlags, lpData)
LPOLEOBJECT lpObject;
UINT wFlags;
LONG FAR* lpData;
```

The ObjectLong function allows the calling application to store data with an object. This function is typically used by object handlers.

Parameter	Description								
lpObject	Points to the <u>OLEOBJECT</u> structure for which the data is stored.								
wFlags	Specifies the method used for setting and retrieving data. It can be one or more of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>OF_SET</td><td>Data is written to the location specified by the lpData parameter, replacing any data already there.</td></tr><tr><td>OF_GET</td><td>Data is read from the location specified by the lpData parameter.</td></tr><tr><td>OF_HANDLER</td><td>Data is written or read by an object handler. This value prevents data from an object handler from being replaced by other applications.</td></tr></tbody></table>	Value	Meaning	OF_SET	Data is written to the location specified by the lpData parameter, replacing any data already there.	OF_GET	Data is read from the location specified by the lpData parameter.	OF_HANDLER	Data is written or read by an object handler. This value prevents data from an object handler from being replaced by other applications.
Value	Meaning								
OF_SET	Data is written to the location specified by the lpData parameter, replacing any data already there.								
OF_GET	Data is read from the location specified by the lpData parameter.								
OF_HANDLER	Data is written or read by an object handler. This value prevents data from an object handler from being replaced by other applications.								
lpData	Points to data to be written or read.								
Returns	The return value is OLE_OK if the function is successful. Otherwise, it is an error value.								
Function									

## SetColorScheme (OLEOBJECTVTBL 1.x)

```
OLESTATUS SetColorScheme(lpObject, lpPal)
LPOLEOBJECT lpObject;
OLE_CONST LOGPALETTE FAR* lpPal;
```

The SetColorScheme function sends the server application the color palette recommended by the client application.

Parameter	Description
lpObject	Points to an <u>OLEOBJECT</u> structure for which the client application recommends a palette.
lpPal	Points to a <u>LOGPALETTE</u> structure specifying the recommended palette.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

Server applications are not required to use the palette recommended by the client application.

Before returning from the SetColorScheme function, the server application should use the palette pointed to by the lpPal parameter in a call to the CreatePalette function to create the handle of the palette:

```
hpal = CreatePalette(lpPal);
```

The server can then use the palette handle to refer to the palette.

The first palette entry in the LOGPALETTE structure specifies the foreground color recommended by the client application. The second palette entry specifies the background color. The first half of the remaining palette entries are fill colors, and the second half are colors for lines and text.

Client applications typically specify an even number of palette entries. When there is an uneven number of entries, the server should interpret the odd entry as a fill color; that is, if there are five entries, three should be interpreted as fill colors and two as line and text colors.

## OLESERVER (3.1)

```
#include <ole.h>
```

```
typedef struct _OLESERVER { /* os */
    LPOLESERVERVTBL lpvtbl;
    . /* any server-supplied state information */
    .
} OLESERVER;
```

The OLESERVER structure points to a table of function pointers for the server. This structure is initialized and maintained by servers for the server library.

Member	Description
lpvtbl	Points to a table of function pointers for the server.

See Also

[OLESERVERVTBL](#)

## OLESERVERDOC (3.1)

```
#include <ole.h>
```

```
typedef struct _OLESERVERDOC { /* osd */
    LPOLESERVERDOCVTBL lpvtbl;
    . /* any server-supplied document-state information */
    .
} OLESERVERDOC;
```

The OLESERVERDOC structure points to a table of function pointers for a document. This structure is initialized and maintained by servers for the server library.

Member	Description
lpvtbl	Points to a table of function pointers for the document.

See Also

[OLESERVERDOCVTBL](#)



### OLESERVERDOCVTBL (3.1)

```
#include <ole.h>

typedef struct _OLESERVERDOCVTBL { /* odv */
    OLESTATUS (CALLBACK* Save)(LPOLESERVERDOC);
    OLESTATUS (CALLBACK* Close)(LPOLESERVERDOC);
    OLESTATUS (CALLBACK* SetHostNames)(LPOLESERVERDOC, OLE_LPCSTR,
    OLE_LPCSTR);
    OLESTATUS (CALLBACK* SetDocDimensions)(LPOLESERVERDOC,
    OLE_CONST RECT FAR*);
    OLESTATUS (CALLBACK* GetObject)(LPOLESERVERDOC, OLE_LPCSTR,
    LPOLEOBJECT FAR*, LPOLECLIENT);
    OLESTATUS (CALLBACK* Release)(LPOLESERVERDOC);
    OLESTATUS (CALLBACK* SetColorScheme)(LPOLESERVERDOC,
    OLE_CONST LOGPALETTE FAR*);
    OLESTATUS (CALLBACK* Execute)(LPOLESERVERDOC, HGLOBAL);
} OLESERVERDOCVTBL;
```

The OLESERVERDOCVTBL structure points to functions that manipulate a document. A server application creates this structure and an OLESERVERDOC structure to give the server library access to a document.

Documents opened or created on request from the library should not be shown to the user for editing until the library requests that they be shown.

Every function except Release can return OLE\_BUSY.

Member	Description
<u>Save</u>	Instructs the server to save the document.
<u>Close</u>	Instructs the server application to unconditionally close the document.
<u>SetHostNames</u>	Sets the names that should be used for window titles.
<u>SetDocDimensions</u>	Gives the server the rectangle on the target device for which the object should be formatted.
<u>GetObject</u>	Requests the server to create an <u>OLEOBJECT</u> structure.
<u>Release</u>	Notifies the server when a revoked document has terminated conversations and may be destroyed.
<u>SetColorScheme</u>	Specifies the color palette preferred by the client application.
<u>Execute</u>	Specifies DDE execute strings.

See Also  
OLESERVERDOC

## Save (OLE 1.x)

OLESTATUS Save(lpDoc)  
LPOLESERVERDOC lpDoc;

The Save function instructs the server to save the document.

Parameter	Description
lpDoc	Points to an <u>OLESERVERDOC</u> structure corresponding to the document to save.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Function

## Close (OLE 1.x)

OLESTATUS Close(lpDoc)  
LPOLESERVERDOC lpDoc;

The Close function instructs the server application to unconditionally close the document. The library calls this function when the client application initiates the closure.

Parameter	Description
lpDoc	Points to an <u>OLESERVERDOC</u> structure corresponding to the document to close.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

The library always calls the Close function before calling the Release function in the OLESERVERVTBL structure.

The server application should not prompt the user to save the document or take other actions; messages of this kind are handled by the client application.

When the library calls the Close function, the server should respond by calling the OleRevokeServerDoc function. The resources for the document are freed when the library calls the Release function. The server should not wait for the Release function by entering a message-dispatch loop after calling OleRevokeServerDoc. (A server should never enter message-dispatch loops while processing any of these functions.)

When a document is closed, the server should free the memory for the OLESERVERDOCVTBL structure and associated resources.

### Function

## SetHostNames (OLE 1.x)

```
OLESTATUS SetHostNames(lpDoc, lpszClient, lpszDoc)
LPOLESERVERDOC lpDoc;
OLE_LPCSTR lpszClient;
OLE_LPCSTR lpszDoc;
```

The SetHostNames function sets the name that should be used for a window title. This name is used only for an embedded object, because a linked object has its own title. This function is used only for documents that are embedded objects.

Parameter	Description
lpDoc	Points to an <u>OLESERVERDOC</u> structure corresponding to a document that is the embedded object for which a name is specified.
lpszClient	Points to a null-terminated string specifying the name of the client.
lpszDoc	Points to a null-terminated string specifying the client's name for the object.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Function

## SetDocDimensions (OLE 1.x)

```
OLESTATUS SetDocDimensions(lpDoc, lpRect)
LPOLESERVERDOC lpDoc;
OLE_CONST RECT FAR* lpRect;
```

The SetDocDimensions function gives the server the rectangle on the target device for which the object should be formatted. This function is relevant only for documents that are embedded objects.

Parameter	Description
lpDoc	Points to the <u>OLESERVERDOC</u> structure corresponding to the document that is the embedded object for which the target size is specified.
lpRect	Points to a <u>RECT</u> structure containing the target size of the object, in <u>MM_HIMETRIC</u> units. (In the <u>MM_HIMETRIC</u> mapping mode, the positive y-direction is up.)

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Function

## GetObject (OLE 1.x)

```
OLESTATUS GetObject(lpDoc, lpszItem, lpObject, lpClient)
LPOLESERVERDOC lpDoc;
OLE_LPCSTR lpszItem;
LPOLEOBJECT FAR* lpObject;
LPOLECLIENT lpClient;
```

The GetObject function requests the server to create an OLEOBJECT structure.

Parameter	Description
lpDoc	Points to an <u>OLESERVERDOC</u> structure corresponding to this document.
lpszItem	Points to a null-terminated string specifying the name of an item in the specified document for which an object structure is requested. If this string is set to NULL, the entire document is requested. This string cannot contain a slash mark (/).
lpObject	Points to a variable of type <u>LPOLEOBJECT</u> in which the server application should return a long pointer to the allocated <u>OLEOBJECT</u> structure.
lpClient	Points to an <u>OLECLIENT</u> structure allocated by the library. The server should associate the <u>OLECLIENT</u> structure with the object and use it to notify the library of changes to the object.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

The server application should allocate and initialize the OLEOBJECT structure, associate it with the OLECLIENT structure pointed to by the lpClient parameter, and return a pointer to the OLEOBJECT structure through the lpObject argument.

The library calls the GetObject function to associate a client with the part of the document identified by the lpszItem parameter. When a client has been associated with an object by this function, the server can send notifications to the client.

Applications should be prepared to handle multiple calls to GetObject for a given object. This entails creating multiple OLECLIENT structures and sending notifications to each of these structures when appropriate. Multiple calls to GetObject are possible because some client applications that implement object linking and embedding (OLE) by using dynamic data exchange (DDE) rather than the OLE dynamic-link libraries may use both NULL and an actual item name for the lpszItem parameter.

### Function

## Release (OLE 1.x)

OLESTATUS Release(lpDoc)  
LPOLESERVERDOC lpDoc;

The Release function notifies the server when a revoked document has terminated conversations and can be destroyed.

Parameter	Description
lpDoc	Points to an <u>OLESERVERDOC</u> structure for which the handle was revoked and which can now be released.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Function

## SetColorScheme (OLESERVERDOCVTBL 1.x)

```
OLESTATUS SetColorScheme(lpDoc, lpPal)
LPOLESERVERDOC lpDoc;
OLE_CONST LOGPALETTE FAR* lpPal;
```

The SetColorScheme function sends the server application the color palette recommended by the client application.

Parameter	Description
lpDoc	Points to an <u>OLESERVERDOC</u> structure for which the client application recommends a palette.
lpPal	Points to a <u>LOGPALETTE</u> structure specifying the recommended palette.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

Server applications are not required to use the palette recommended by the client application.

Before returning from the SetColorScheme function, the server application should create a handle to the palette. To do this, the server application should use the palette pointed to by the lpPal parameter in a call to the CreatePalette function, as shown in the following example.

```
hpal = CreatePalette(lpPal);
```

The server can then use the palette handle to refer to the palette.

The first palette entry in the LOGPALETTE structure specifies the foreground color recommended by the client application. The second palette entry specifies the background color. The first half of the remaining palette entries are fill colors, and the second half are colors for lines and text.

Client applications typically specify an even number of palette entries. When there is an uneven number of entries, the server should interpret the odd entry as a fill color; that is, if there are five entries, three should be interpreted as fill colors and two as line and text colors.

### Function



## Execute (OLE 1.x)

OLESTATUS Execute(lpDoc, hCommands)  
LPOLESERVERDOC lpDoc;  
HGLOBAL hCommands;

The Execute function receives WM\_DDE\_EXECUTE commands sent by client applications. The applications send these commands by calling the OleExecute function.

Parameter	Description
lpDoc	Points to an <u>OLESERVERDOC</u> structure to which the dynamic data exchange (DDE) commands apply.
hCommands	Identifies memory containing one or more DDE execute commands.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

The server should never free the handle specified in the hCommands parameter.

### OLESERVERVTBL (3.1)

```
#include <ole.h>

typedef struct _OLESERVERVTBL { /* osv */
    OLESTATUS (CALLBACK* Open)(LPOLESERVER, LHSERVERDOC,
    OLE_LPCSTR, LPOLESERVERDOC FAR*);
    OLESTATUS (CALLBACK* Create)(LPOLESERVER, LHSERVERDOC,
    OLE_LPCSTR, OLE_LPCSTR, LPOLESERVERDOC FAR*);
    OLESTATUS (CALLBACK* CreateFromTemplate)(LPOLESERVER,
    LHSERVERDOC, OLE_LPCSTR, OLE_LPCSTR, OLE_LPCSTR,
    LPOLESERVERDOC FAR*);
    OLESTATUS (CALLBACK* Edit)(LPOLESERVER, LHSERVERDOC,
    OLE_LPCSTR, OLE_LPCSTR, LPOLESERVERDOC FAR*);
    OLESTATUS (CALLBACK* Exit)(LPOLESERVER);
    OLESTATUS (CALLBACK* Release)(LPOLESERVER);
    OLESTATUS (CALLBACK* Execute)(LPOLESERVER, HGLOBAL);
} OLESERVERVTBL;
```

The OLESERVERVTBL structure points to functions that manipulate a server. After a server application creates this structure and an OLESERVER structure, the server library can perform operations on the server application.

Every function except Release can return OLE\_BUSY.

Member	Description
<u>Open</u>	Opens an existing file and prepares to edit the contents.
<u>Create</u>	Makes a new object of a given class name which will be embedded in the client application.
<u>CreateFromTemplate</u>	Creates a new document that is initialized with the data in a specified file.
<u>Edit</u>	Creates a document that is initialized with data retrieved by a subsequent call to the <u>SetData</u> function.
<u>Exit</u>	Instructs the server application to close documents and shut down.
<u>Release</u>	Notifies a server that all connections to it have closed and that it is safe to terminate.
<u>Execute</u>	Specifies DDE execute strings.

Function

See Also

OLESERVER

## Open (OLE 1.x)

```
OLESTATUS Open(lpServer, lhDoc, lpszDoc, lpDoc)  
LPOLESERVER lpServer;  
LHSERVERDOC lhDoc;  
OLE_LPCSTR lpszDoc;  
LPOLESERVERDOC FAR* lpDoc;
```

The Open function opens an existing file and prepares to edit the contents. A server typically uses this function to open a linked object for a client application.

Parameter	Description
lpServer	Points to an <u>OLESERVER</u> structure identifying the server.
lhDoc	Identifies the document. The library uses this handle internally.
lpszDoc	Points to a null-terminated string specifying the permanent name of the document to be opened. Typically this string is a path, but for some applications it might be further qualified. For example, the string might specify a particular table in a database.
lpDoc	Points to a variable of type <u>LPOLESERVERDOC</u> in which the server application returns a long pointer to the <u>OLESERVERDOC</u> structure it has created in response to this function.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

When the library calls this function, the server application opens a specified document, allocates and initializes an OLESERVERDOC structure, associates the library's handle with the document, and returns the address of the structure. The server does not show the document or its window.

### Function

## Create (OLE 1.x)

```
OLESTATUS Create(lpServer, lhDoc, lpszClass, lpszDoc, lpDoc)
LPOLESERVER lpServer;
LHSERVERDOC lhDoc;
OLE_LPCSTR lpszClass;
OLE_LPCSTR lpszDoc;
LPOLESERVERDOC FAR* lpDoc;
```

The Create function makes a new object that is to be embedded in the client application. The lpszDoc parameter identifies the object but should not be used to create a file for the object.

Parameter	Description
lpServer	Points to an <u>OLESERVER</u> structure identifying the server.
lhDoc	Identifies the document. The library uses this handle internally.
lpszClass	Points to a null-terminated string specifying the class of document to create.
lpszDoc	Points to a null-terminated string specifying a name for the document to be created. This name can be used to identify the document in window titles.
lpDoc	Points to a variable of type <u>LPOLESERVERDOC</u> in which the server application should return a long pointer to the created <u>OLESERVERDOC</u> structure.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

When the library calls this function, the server application creates a document of a specified class, allocates and initializes an OLESERVERDOC structure, associates the library's handle with the document, and returns the address of the structure. This function opens the created document for editing and embeds it in the client when it is updated or closed.

Server applications often track changes to the document specified in this function, so that the user can be prompted to save changes when necessary.

### Function

## CreateFromTemplate (OLE 1.x)

```
OLESTATUS CreateFromTemplate(lpServer, lhDoc, lpszClass, lpszDoc, lpszTemplate, lpDoc)
LPOLESERVER lpServer;
LHSERVERDOC lhDoc;
OLE_LPCSTR lpszClass;
OLE_LPCSTR lpszDoc;
OLE_LPCSTR lpszTemplate;
LPOLESERVERDOC FAR* lpDoc;
```

The CreateFromTemplate function creates a new document that is initialized with the data in a specified file. The new document is opened for editing by this function and embedded in the client when it is updated or closed.

Parameter	Description
lpServer	Points to an <u>OLESERVER</u> structure identifying the server.
lhDoc	Identifies the document. The library uses this handle internally.
lpszClass	Points to a null-terminated string specifying the class of document to create.
lpszDoc	Points to a null-terminated string specifying a name for the document to be created. This name need not be used by the server application but can be used in window titles.
lpszTemplate	Points to a null-terminated string specifying the permanent name of the document to use to initialize the new document. Typically this string is a path, but for some applications it might be further qualified. For example, the string might specify a particular table in a database.
lpDoc	Points to a variable of type <u>LPOLESERVERDOC</u> in which the server application should return a long pointer to the created <u>OLESERVERDOC</u> structure.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

When the library calls this function, the server application creates a document of a specified class, allocates and initializes an OLESERVERDOC structure, associates the library's handle with the document, and returns the address of the structure.

A server application often tracks changes to the document specified in this function, so that the user can be prompted to save changes when necessary.

### Function

## Edit (OLE 1.x)

```
OLESTATUS Edit(lpServer, lhDoc, lpszClass, lpszDoc, lpDoc)
LPOLESERVER lpServer;
LHSERVERDOC lhDoc;
OLE_LPCSTR lpszClass;
OLE_LPCSTR lpszDoc;
LPOLESERVERDOC FAR* lpDoc;
```

The Edit function creates a document that is initialized with data retrieved by a subsequent call to the SetData function. The object is embedded in the client application. The server does not show the document or its window.

Parameter	Description
lpServer	Points to an <u>OLESERVER</u> structure identifying the server.
lhDoc	Identifies the document. The library uses this handle internally.
lpszClass	Points to a null-terminated string specifying the class of document to create.
lpszDoc	Points to a null-terminated string specifying a name for the document to be created. This name need not be used by the server application but may be used--for example, in a window title.
lpDoc	Points to a variable of type <u>LPOLESERVERDOC</u> in which the server application should return a long pointer to the created <u>OLESERVERDOC</u> structure.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

When the library calls this function, the server application creates a document of a specified class, allocates and initializes an OLESERVERDOC structure, associates the library's handle with the document, and returns the address of the structure.

The document created by the Edit function retrieves the initial data from the client in a subsequent call to the SetData function. The user can edit the document after the data has been retrieved and the library has used either the Show function in the OLEOBJECTVTBL structure or the DoVerb function with an Edit verb to show the document to the user.

### Function

## Exit (OLE 1.x)

OLESTATUS Exit(lpServer)  
LPOLESERVER lpServer;

The Exit function instructs the server application to close documents and quit.

Parameter	Description
lpServer	Points to an <u>OLESERVER</u> structure identifying the server.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

The server library calls the Exit function to instruct a server application to terminate. If the server application has no open documents when the Exit function is called, it should call the OleRevokeServer function.

### Function

## Release (OLE 1.x)

OLESTATUS Release(lpServer)  
LPOLESERVER lpServer;

The Release function notifies a server that all connections to it have closed and that it is safe to quit.

Parameter	Description
lpServer	Points to an <u>OLESERVER</u> structure identifying the server.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

The server library calls the Release function when it is safe for a server to quit. When a server application calls the OleRevokeServer function, the application must continue to dispatch messages and wait for the library to call the Release function before quitting.

When the server is invisible and the library calls Release, the server must exit. (The only exception is when an application supports multiple servers; in this case, an invisible server is sometimes not revocable when the library calls Release.) If the server has no open documents and it was started with the / Embedding option (indicating that it was started by a client application), the server should exit when the library calls the Release function. If the user has explicitly loaded a document into a single-instance multiple document interface server, however, the server should not exit when the library calls Release. Typically, a single-instance server is a multiple document interface (MDI) server.

All registered server structures must be released before a server can quit.

A server can call the PostQuitMessage function inside the Release function.

### Function

### See Also

PostQuitMessage



## Execute (OLE 1.x)

OLESTATUS Execute(lpServer, hCommands)  
LPOLESERVER lpServer;  
HGLOBAL hCommands;

The Execute function receives WM\_DDE\_EXECUTE commands sent by client applications. The applications send these commands by calling the OleExecute function.

Parameter	Description
lpServer	Points to an <u>OLESERVER</u> structure identifying the server.
hCommands	Identifies memory containing one or more dynamic data exchange (DDE) execute commands.

### Returns

The return value is OLE\_OK if the function is successful. Otherwise, it is an error value.

### Comments

The server should never free the handle specified in the hCommands parameter.

## OLESTREAM (3.1)

```
#include <ole.h>
```

```
typedef struct _OLESTREAM { /* ostr */  
    LPOLESTREAMVTBL lpstbl;  
} OLESTREAM;
```

The OLESTREAM structure points to an OLESTREAMVTBL structure that provides stream input and output functions. These functions are used by the client library for stream operations on objects. The OLESTREAM structure is allocated and initialized by client applications.

Member	Description
lpstbl	Points to an <u>OLESTREAMVTBL</u> structure.

See Also

OLESTREAMVTBL

## OLESTREAMVTBL (3.1)

```
#include <ole.h>
```

```
typedef struct _OLESTREAMVTBL { /* ostrv */  
    DWORD (CALLBACK* Get)(LPOLESTREAM, void FAR*, DWORD);  
    DWORD (CALLBACK* Put)(LPOLESTREAM, OLE_CONST void FAR*, DWORD);  
} OLESTREAMVTBL;
```

The OLESTREAMVTBL structure points to functions the client library uses for stream operations on objects. This structure is allocated and initialized by client applications.

Member	Description
<b><u>Get</u></b>	Gets data from the stream.
<b><u>Put</u></b>	Puts data into the stream.

### Comments

The stream is valid only for the duration of the function to which it is passed. The library obtains everything it requires while the stream is valid.

The return values for the stream functions may indicate that an error has occurred, but these values do not indicate the nature of the error. The client application is responsible for any required error-recovery operations.

A client application can use these functions to provide variations on the standard stream procedures; for example, the client could change the permanent storage of some objects so that they were stored in a database instead of the client document.

### Function

See Also

[OLESTREAM](#)

## Get (OLE 1.x)

```
DWORD Get(lpstream, lpszBuf, cbbuf)
LPOLESTREAM lpstream;
void FAR* lpszBuf;
DWORD cbbuf;
```

The Get function gets data from the specified stream.

Parameter	Description
lpstream	Points to an <u>OLESTREAM</u> structure allocated by the client.
lpszBuf	Points to a buffer to fill with data from the stream.
cbbuf	Specifies the number of bytes to read into the buffer.

### Returns

The return value is the number of bytes actually read into the buffer if the function is successful. If the end of the file is encountered, the return value is zero. A negative return value indicates that an error occurred.

### Comments

The value specified by the cbbuf parameter can be larger than 64K. If the client application uses a stream-reading function that is limited to 64K, it should call that function repeatedly until it has read the number of bytes specified by cbbuf. Whenever the data size is larger than 64K, the pointer to the data buffer is always at the beginning of the segment.

### Function

## Put (OLE 1.x)

```
DWORD Put(lpstream, lpszBuf, cbbuf)
LPOLESTREAM lpstream;
OLE_CONST void FAR* lpszBuf;
DWORD cbbuf;
```

The Put function puts data into the specified stream.

Parameter	Description
lpstream	Points to an <u>OLESTREAM</u> structure allocated by the client.
lpszBuf	Points to a buffer from which to write data into the stream.
cbbuf	Specifies the number of bytes to write into the stream.

### Returns

The return value is the number of bytes actually written to the stream. A return value less than the number specified in the cbbuf parameter indicates that either there was insufficient space in the stream or an error occurred.

### Comments

The value specified by the cbbuf parameter can be greater than 64K. If the client application uses a stream-writing function that is limited to 64K, it should call that function repeatedly until it has written the number of bytes specified by cbbuf. Whenever the data size is greater than 64K, the pointer to the data buffer is always at the beginning of the segment.

## OLETARGETDEVICE (3.1)

```
#include <ole.h>

typedef struct _OLETARGETDEVICE {    /* otd */
    UINT otdDeviceNameOffset;
    UINT otdDriverNameOffset;
    UINT otdPortNameOffset;
    UINT otdExtDevmodeOffset;
    UINT otdExtDevmodeSize;
    UINT otdEnvironmentOffset;
    UINT otdEnvironmentSize;
    BYTE otdData[1];
} OLETARGETDEVICE;
```

The OLETARGETDEVICE structure contains information about the target device that a client application is using. Server applications can use the information in this structure to change the rendering of an object, if necessary. A client application provides a handle to this structure in a call to the OleSetTargetDevice function.

Member	Description
otdDeviceNameOffset	Specifies the offset from the beginning of the array to the name of the device.
otdDriverNameOffset	Specifies the offset from the beginning of the array to the name of the device driver.
otdPortNameOffset	Specifies the offset from the beginning of the array to the name of the port.
otdExtDevmodeOffset	Specifies the offset from the beginning of the array to a <u>DEVMODE</u> structure retrieved by the <u>ExtDeviceMode</u> function.
otdExtDevmodeSize	Specifies the size of the <u>DEVMODE</u> structure whose offset is specified by the otdExtDevmodeOffset member.
otdEnvironmentOffset	Specifies the offset from the beginning of the array to the device environment.
otdEnvironmentSize	Specifies the size of the environment whose offset is specified by the otdEnvironmentOffset member.
otdData	Specifies an array of bytes containing data for the target device.

### Comments

The otdDeviceNameOffset, otdDriverNameOffset, and otdPortNameOffset members should be NULL-terminated.

In Windows 3.1, the ability to connect multiple printers to one port has made the environment obsolete. The environment information retrieved by the GetEnvironment function can occasionally be incorrect. To ensure that the OLETARGETDEVICE structure is initialized correctly, the application should copy information from the DEVMODE structure retrieved by a call to the ExtDeviceMode function to the environment position of the OLETARGETDEVICE structure.

### See Also

OleSetTargetDevice

## OPENFILENAME (3.1)

```
#include <commdlg.h>

typedef struct tagOPENFILENAME { /* ofn */
    DWORD lStructSize;
    HWND hwndOwner;
    HINSTANCE hInstance;
    LPCSTR lpstrFilter;
    LPSTR lpstrCustomFilter;
    DWORD nMaxCustFilter;
    DWORD nFilterIndex;
    LPSTR lpstrFile;
    DWORD nMaxFile;
    LPSTR lpstrFileName;
    DWORD nMaxFileName;
    LPCSTR lpstrInitialDir;
    LPCSTR lpstrTitle;
    DWORD flags;
    UINT nFileOffset;
    UINT nFileExtension;
    LPCSTR lpstrDefExt;
    LPARAM lCustData;
    UINT (CALLBACK* lpfnHook) (HWND, UINT, WPARAM, LPARAM);
    LPCSTR lpTemplateName;
} OPENFILENAME;
```

The OPENFILENAME structure contains information that the system uses to initialize the system-defined Open dialog box or Save dialog box. After the user chooses the OK button to close the dialog box, the system returns information about the user's selection in this structure.

Member	Description
lStructSize	Specifies the length of the structure, in bytes. This member is filled on input.
hwndOwner	Identifies the window that owns the dialog box. This member can be any valid window handle, or it should be NULL if the dialog box is to have no owner.  If the <code>OFN_SHOWHELP</code> flag is set, <code>hwndOwner</code> must identify the window that owns the dialog box. The window procedure for this owner window receives a notification message when the user chooses the Help button. (The identifier for the notification message is the value returned by the <a href="#">RegisterWindowMessage</a> function when <code>HELPMMSGSTRING</code> is passed as its argument.)
hInstance	This member is filled on input.  Identifies a data block that contains a dialog box template specified by the <code>lpTemplateName</code> member. This member is used only if the <code>Flags</code> member specifies the <code>OFN_ENABLETEMPLATE</code> or the <code>OFN_ENABLETEMPLATEHANDLE</code> flag; otherwise, this member is ignored.
lpstrFilter	This member is filled on input.  Points to a buffer containing one or more pairs of null-terminated strings specifying filters. The first string in each pair describes a filter (for example, "Text Files"); the second specifies the filter pattern (for example, "*.txt"). Multiple filters can be specified for a single item; in this case, the semicolon (;) is used to separate filter pattern strings--for example, "*.txt*.doc*.bak". The last string in the buffer must be terminated by two null characters. If this parameter is NULL, the dialog box does not display any filters. The filter strings must be in the proper order--the system does not change the order.
lpstrCustomFilter	This member is filled on input.  Points to a buffer containing a pair of user-defined strings that specify a filter. These strings should be formatted in the same way as is described for the <code>lpstrFilter</code> member. The first string describes the filter and the second specifies the filter pattern (for example, "WinWord", "*.doc"). The buffer is terminated by two null characters. The system copies the strings from the File Name edit control to the buffer when the user chooses the OK button to close the dialog box.

	<p>If the <code>nFilterIndex</code> member is zero, the system uses the <code>lpstrCustomFilter</code> strings as the initial filter description and filter pattern for the dialog box. If the first string in the pair pointed to by <code>lpstrCustomFilter</code> is NULL (for example, "", "*.doc"), only the strings pointed to by <code>lpstrFilter</code> are displayed in the List Files of Type listbox, but the last specified filter pattern is still passed to the second string location of <code>lpstrCustomFilter</code>.</p>
<code>nMaxCustFilter</code>	<p>Specifies the size, in bytes, of the buffer identified by the <code>lpstrCustomFilter</code> member. This buffer should be at least 40 bytes long. This parameter is ignored if the <code>lpstrCustomFilter</code> member is NULL.</p>
<code>nFilterIndex</code>	<p>This member is filled on input.</p> <p>Specifies an index into the buffer pointed to by the <code>lpstrFilter</code> member. The system uses the index value to obtain a pair of strings to use as the initial filter description and filter pattern for the dialog box. The first pair of strings has an index value of 1. When the user chooses the OK button to close the dialog box, the system copies the index of the selected filter strings into this location. If the <code>nFilterIndex</code> member is 0, the filter in the buffer pointed to by the <code>lpstrCustomFilter</code> member is used. If the <code>nFilterIndex</code> member is 0 and the <code>lpstrCustomFilter</code> member is NULL, the system uses the first filter in the buffer pointed to by the <code>lpstrFilter</code> member. If each of the three members is either 0 or NULL, the system does not use any filters and does not show any files in the File Name list box of the dialog box.</p>
<code>lpstrFile</code>	<p>Points to a buffer that specifies a filename used to initialize the File Name edit control. If initialization is not necessary, the first character of this buffer must be NULL. When the <code>GetOpenFileName</code> or <code>GetSaveFileName</code> function returns, this buffer contains the complete location and name of the selected file.</p> <p>If the buffer is too small, the dialog box procedure copies the required size into this member and returns 0. To retrieve the required size, cast the <code>lpstrFile</code> member to type <code>LPWORD</code>. The buffer must be at least three bytes to receive the required size. When the buffer is too small, the <code>CommDlgExtendedError</code> function returns the <code>FNERR_BUFFERTOOSMALL</code> value.</p>
<code>nMaxFile</code>	<p>Specifies the size, in bytes, of the buffer pointed to by the <code>lpstrFile</code> member. The <code>GetOpenFileName</code> and <code>GetSaveFileName</code> functions return FALSE if the buffer is too small to contain the file information. The buffer does not need to be larger than 128 bytes; <code>lpstrFile</code> entries longer than 128 bytes are truncated. If the <code>lpstrFile</code> member is NULL, this member is ignored.</p>
<code>lpstrFileTitle</code>	<p>This member is filled on input.</p> <p>Points to a buffer that receives the title of the selected file. This buffer receives the filename and extension but no path information. An application should use this string to display the file title. If this member is NULL, the function does not copy the file title. This member is filled on output.</p>
<code>nMaxFileTitle</code>	<p>Specifies the maximum length, in bytes, of the string that can be copied into the <code>lpstrFileTitle</code> buffer. This member is ignored if <code>lpstrFileTitle</code> is NULL. This member is filled on input.</p>
<code>lpstrInitialDir</code>	<p>Points to a string that specifies the initial file directory. If this member is NULL, the system uses the current directory as the initial directory. (If the <code>lpstrFile</code> member contains a string that specifies a valid path, the common dialog box procedure will use the path specified by this string instead of the path specified by the string to which <code>lpstrInitialDir</code> points.)</p> <p>This member is filled on input.</p>
<code>lpstrTitle</code>	<p>Points to a string to be placed in the title bar of the dialog box. If this member is NULL, the system uses the default title (that is, Save As or Open). This member is filled on input.</p>
Flags	<p>Specifies the dialog box initialization flags. This member may be a combination of the following values:</p> <p>Value Meaning</p> <p><b><u>OFN_ALLOWMULTISELECT</u></b>  Specifies that the File Name list box is to allow multiple selections. When this flag is set, the <code>lpstrFile</code> member points to a buffer containing the path to the current directory and all filenames in the selection. The first filename is separated from the path by a space. Each subsequent filename is separated by one space from the preceding filename. Some of</p>



the selected filenames may be preceded by relative paths; for example, the buffer could contain something like this:

```
c:\files file1.txt file2.txt ..\bin\file3.txt
```

**OFN\_CREATEPROMPT**

Causes the dialog box procedure to generate a message box to notify the user when a specified file does not currently exist and to make it possible for the user to specify that the file should be created. (This flag automatically sets the OFN\_PATHMUSTEXIST and OFN\_FILEMUSTEXIST flags.)

**OFN\_ENABLEHOOK**

Enables the hook function specified in the lpfnHook member.

**OFN\_ENABLETEMPLATE**

Causes the system to use the dialog box template identified by the hInstance and lpTemplateName members to create the dialog box.

**OFN\_ENABLETEMPLATEHANDLE**

Indicates that the hInstance member identifies a data block that contains a pre-loaded dialog box template. The system ignores the lpTemplateName member if this flag is specified.

**OFN\_EXTENSIONDIFFERENT**

Indicates that the extension of the returned filename is different from the extension specified by the lpstrDefExt member. This flag is not set if lpstrDefExt is NULL, if the extensions match, or if the file has no extension. This flag can be set on output.

**OFN\_FILEMUSTEXIST**

Specifies that the user can type only the names of existing files in the File Name edit control. If this flag is set and the user types an invalid filename in the File Name edit control, the dialog box procedure displays a warning in a message box. (This flag also causes the OFN\_PATHMUSTEXIST flag to be set.)

**OFN\_HIDEREADONLY**

Hides the Read Only check box.

**OFN\_NOCHANGEDIR**

Forces the dialog box to reset the current directory to what it was when the dialog box was created.

**OFN\_NOREADONLYRETURN**

Specifies that the file returned will not have the Read Only attribute set and will not be in a write-protected directory.

**OFN\_NOTESTFILECREATE**

Specifies that the file will not be created before the dialog box is closed. This flag should be set if the application saves the file on a create-no-modify network share point. When an application sets this flag, the library does not check against write protection, a full disk, an open drive door, or network protection. Therefore, applications that use this flag must perform file operations carefully--a file cannot be reopened once it is closed.

**OFN\_NOVALIDATE**

Specifies that the common dialog boxes will allow invalid characters in the returned filename. Typically, the calling application uses a hook function that checks the filename using the FILEOKSTRING registered message. If the text in the edit control is empty or contains nothing but spaces, the lists of files and directories are updated. If the text in the edit control contains anything else, the nFileOffset and nFileExtension members are set to values generated by parsing the text. No default extension is added to the text, nor is text copied to the lpstrFileName buffer.

If the value specified by the nFileOffset member is negative, the filename is invalid. If the value specified by nFileOffset is not negative, the filename is valid, and nFileOffset and nFileExtension can be used as if the OFN\_NOVALIDATE flag had not been set.

#### OFN\_OVERWRITEPROMPT

Causes the Save As dialog box to generate a message box if the selected file already exists. The user must confirm whether to overwrite the file.

#### OFN\_PATHMUSTEXIST

Specifies that the user can type only valid paths. If this flag is set and the user types an invalid path in the File Name edit control, the dialog box procedure displays a warning in a message box.

#### OFN\_READONLY

Causes the Read Only check box to be initially checked when the dialog box is created. When the user chooses the OK button to close the dialog box, the state of the Read Only check box is specified by this member. This flag can be set on input and output.

#### OFN\_SHAREAWARE

Specifies that if a call to the OpenFile function has failed because of a network sharing violation, the error is ignored and the dialog box returns the given filename. If this flag is not set, the registered message for SHAREVISTRING is sent to the hook function, with a pointer to a null-terminated string for the path name in the IParam parameter. The hook function responds with one of the following values:

Value      Meaning

#### OFN\_SHAREFALLTHROUGH

Specifies that the filename is returned from the dialog box.

#### OFN\_SHARENOWARN

Specifies no further action.

#### OFN\_SHAREWARN

Specifies that the user receives the standard warning message for this error. (This is the same result as if there were no hook function.)

This flag may be set on output.

#### OFN\_SHOWHELP

Causes the dialog box to show the Help push button. The hwndOwner must not be NULL if this option is specified.

These flags may be set when the structure is initialized, except where specified.

nFileOffset

Specifies a zero-based offset from the beginning of the path to the filename specified by the string in the buffer to which lpstrFile points. For example, if lpstrFile points to the string, "c:\dir1\dir2\file.ext", this member contains the value 13.

This member is filled on output.

nFileExtension

Specifies a zero-based offset from the beginning of the path to the filename extension specified by the string in the buffer to which lpstrFile points. For example, if lpstrFile points to the following string, "c:\dir1\dir2\file.ext", this member contains the value 18. If the user did not type an extension and lpstrDefExt is NULL, this member specifies an offset to the terminating null character. If the user typed a period (.) as the last character in the filename, this member is 0.

This member is filled on output.

lpstrDefExt

Points to a buffer that contains the default extension. The GetOpenFileName or GetSaveFileName function appends this extension to the filename if the user fails to enter an extension. If the filename with the default extension is not found, the GetOpenFileName or GetSaveFileName function attempts to find the file using the name exactly as the user typed it. This string can be any length, but only the first three characters are appended. The string should not contain a period (.). If this member is NULL and the user fails to type an extension, no extension is appended. This member is filled on input.

lCustData

Specifies application-defined data that the system passes to the hook function pointed to by the lpfnHook member. The system passes a pointer to the OPENFILENAME structure in the IParam parameter of the WM\_INITDIALOG message; this pointer can be used to retrieve the lCustData member.

**lpfnHook** Points to a hook function that processes messages intended for the dialog box. To enable the hook function, an application must specify the OFN\_ENABLEHOOK flag in the **Flags** member; otherwise, the system ignores this structure member. The hook function must return zero to pass a message that it didn't process back to the dialog box procedure in **COMMDLG.DLL**. The hook function must return a nonzero value to prevent the dialog box procedure in **COMMDLG.DLL** from processing a message it has already processed.  
This member is filled on input.

**lpTemplateName** Points to a null-terminated string that specifies the name of the resource file for the dialog box template that is to be substituted for the dialog box template in **COMMDLG.DLL**. An application can use the MAKEINTRESOURCE macro for numbered dialog box resources. This member is used only if the **Flags** member specifies the OFN\_ENABLETEMPLATE flag; otherwise, this member is ignored.  
This member is filled on input.

See Also

GetOpenFileName, GetSaveFileName, MAKEINTRESOURCE

OFN\_ALLOWMULTISELECT 0x00000200

Specifies that the File Name list box is to allow multiple selections. When this flag is set, the lpstrFile member points to a buffer containing the path to the current directory and all filenames in the selection. The first filename is separated from the path by a space. Each subsequent filename is separated by one space from the preceding filename. Some of the selected filenames may be preceded by relative paths; for example, the buffer could contain something like this:

OFN\_ALLOWMULTISELECT 0x00000200

OFN\_CREATEPROMPT 0x00002000

Causes the dialog box procedure to generate a message box to notify the user when a specified file does not currently exist and to make it possible for the user to specify that the file should be created. (This flag automatically sets the OFN\_PATHMUSTEXIST and OFN\_FILEMUSTEXIST flags.)

OFN\_CREATEPROMPT 0x00002000

OFN\_ENABLEHOOK 0x00000020

Enables the hook function specified in the lpfnHook member.



OFN\_ENABLEHOOK 0x00000020

OFN\_ENABLETEMPLATE 0x00000040

Causes the system to use the dialog box template identified by the hInstance and lpTemplateName members to create the dialog box.

OFN\_ENABLETEMPLATE 0x00000040

OFN\_ENABLETEMPLATEHANDLE 0x00000080

Indicates that the hInstance member identifies a data block that contains a pre-loaded dialog box template. The system ignores the lpTemplateName member if this flag is specified.

OFN\_ENABLETEMPLATEHANDLE 0x00000080

OFN\_EXTENSIONDIFFERENT 0x00000400

Indicates that the extension of the returned filename is different from the extension specified by the lpstrDefExt member. This flag is not set if lpstrDefExt is NULL, if the extensions match, or if the file has no extension. This flag can be set on output.

OFN\_EXTENSIONDIFFERENT 0x00000400

OFN\_FILEMUSTEXIST 0x00001000

Specifies that the user can type only the names of existing files in the File Name edit control. If this flag is set and the user types an invalid filename in the File Name edit control, the dialog box procedure displays a warning in a message box. (This flag also causes the OFN\_PATHMUSTEXIST flag to be set.)



OFN\_FILEMUSTEXIST 0x00001000

OFN\_HIDEREADONLY 0x00000004

Hides the Read Only check box.

OFN\_HIDEREADONLY 0x00000004

OFN\_NOCHANGEDIR 0x00000008

Forces the dialog box to reset the current directory to what it was when the dialog box was created.

OFN\_NOCHANGEDIR 0x00000008

OFN\_NOREADONLYRETURN 0x00008000

Specifies that the file returned will not have the Read Only attribute set and will not be in a write-protected directory.

OFN\_NOREADONLYRETURN 0x00008000

OFN\_NOTESTFILECREATE 0x00010000

Specifies that the file will not be created before the dialog box is closed. This flag should be set if the application saves the file on a create-no-modify network share point. When an application sets this flag, the library does not check against write protection, a full disk, an open drive door, or network protection. Therefore, applications that use this flag must perform file operations carefully--a file cannot be reopened once it is closed.



OFN\_NOTESTFILECREATE 0x00010000

OFN\_NOVALIDATE 0x00000100

Specifies that the common dialog boxes will allow invalid characters in the returned filename. Typically, the calling application uses a hook function that checks the filename using the FILEOKSTRING registered message. If the text in the edit control is empty or contains nothing but spaces, the lists of files and directories are updated. If the text in the edit control contains anything else, the nFileOffset and nFileExtension members are set to values generated by parsing the text. No default extension is added to the text, nor is text copied to the lpstrFileName buffer. If the value specified by the nFileOffset member is negative, the filename is invalid. If the value specified by nFileOffset is not negative, the filename is valid, and nFileOffset and nFileExtension can be used as if the OFN\_NOVALIDATE flag had not been set.

OFN\_NOVALIDATE 0x00000100

OFN\_OVERWRITEPROMPT 0x00000002

Causes the Save As dialog box to generate a message box if the selected file already exists. The user must confirm whether to overwrite the file.

OFN\_OVERWRITEPROMPT 0x00000002

OFN\_PATHMUSTEXIST 0x00000800

Specifies that the user can type only valid paths. If this flag is set and the user types an invalid path in the File Name edit control, the dialog box procedure displays a warning in a message box.

OFN\_PATHMUSTEXIST 0x00000800

OFN\_READONLY 0x00000001

Causes the Read Only check box to be initially checked when the dialog box is created. When the user chooses the OK button to close the dialog box, the state of the Read Only check box is specified by this member. This flag can be set on input and output.



OFN\_READONLY 0x00000001

OFN\_SHAREAWARE 0x00004000

Specifies that if a call to the OpenFile function has failed because of a network sharing violation, the error is ignored and the dialog box returns the given filename. If this flag is not set, the registered message for SHAREVISTRING is sent to the hook function, with a pointer to a null-terminated string for the path name in the lParam parameter. The hook function responds with one of the following values:

OFN\_SHAREAWARE 0x00004000

This  
flag may be set on output.

OFN\_SHOWHELP 0x00000010

Causes the dialog box to show the Help push button. The hWndOwner must not be NULL if this option is specified.

OFN\_SHOWHELP 0x00000010

## OUTLINETEXTMETRIC (3.1)

```
typedef struct tagOUTLINETEXTMETRIC {    /* otm */
    UINT    otmSize;
    TEXTMETRIC    otmTextMetrics;
    BYTE    otmFiller;
    PANOSE    otmPanoseNumber;
    UINT    otmfsSelection;
    UINT    otmfsType;
    UINT    otmsCharSlopeRise;
    UINT    otmsCharSlopeRun;
    UINT    otmItalicAngle;
    UINT    otmEMSsquare;
    INT    otmAscent;
    INT    otmDescent;
    UINT    otmLineGap;
    UINT    otmsXHeight;
    UINT    otmsCapEmHeight;
    RECT    otmrcFontBox;
    INT    otmMacAscent;
    INT    otmMacDescent;
    UINT    otmMacLineGap;
    UINT    otmusMinimumPPEM;
    POINT    otmptSubscriptSize;
    POINT    otmptSubscriptOffset;
    POINT    otmptSuperscriptSize;
    POINT    otmptSuperscriptOffset;
    UINT    otmsStrikeoutSize;
    INT    otmsStrikeoutPosition;
    INT    otmsUnderscorePosition;
    UINT    otmsUnderscoreSize;
    PSTR    otmpFamilyName;
    PSTR    otmpFaceName;
    PSTR    otmpStyleName;
    PSTR    otmpFullName;
} OUTLINETEXTMETRIC;
```

The OUTLINETEXTMETRIC structure contains metrics describing a TrueType font.

Member	Description														
otmSize	Specifies the size, in bytes, of the OUTLINETEXTMETRIC structure.														
otmTextMetrics	Specifies a <u>TEXTMETRIC</u> structure containing further information about the font.														
otmFiller	Specifies a value that causes the structure to be byte-aligned.														
otmPanoseNumber	Specifies the Panose number for this font.														
otmfsSelection	Specifies the nature of the font pattern. This member can be a combination of the following bits: <table border="1" data-bbox="636 1465 841 1690"> <thead> <tr> <th>Bit</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Italic</td> </tr> <tr> <td>1</td> <td>Underscore</td> </tr> <tr> <td>2</td> <td>Negative</td> </tr> <tr> <td>3</td> <td>Outline</td> </tr> <tr> <td>4</td> <td>Strikeout</td> </tr> <tr> <td>5</td> <td>Bold</td> </tr> </tbody> </table>	Bit	Meaning	0	Italic	1	Underscore	2	Negative	3	Outline	4	Strikeout	5	Bold
Bit	Meaning														
0	Italic														
1	Underscore														
2	Negative														
3	Outline														
4	Strikeout														
5	Bold														
otmfsType	Specifies whether the font is licensed. Licensed fonts may not be modified or exchanged. If bit 1 is set, the font may not be embedded in a document. If bit 1 is clear, the font can be embedded. If bit 2 is set, the embedding is read-only.														
otmsCharSlopeRise	Specifies the slope of the cursor. This value is 1 if the slope is vertical. Applications can use this value and the value of the otmsCharSlopeRun member to create an italic cursor that has the same														

	slope as the main italic angle (specified by the <code>otmItalicAngle</code> member).
<code>otmsCharSlopeRun</code>	Specifies the slope of the cursor. This value is zero if the slope is vertical. Applications can use this value and the value of the <code>otmsCharSlopeRise</code> member to create an italic cursor that has the same slope as the main italic angle (specified by the <code>otmItalicAngle</code> member).
<code>otmItalicAngle</code>	Specifies the main italic angle of the font, in counterclockwise degrees from vertical. Regular (roman) fonts have a value of zero. Italic fonts typically have a negative italic angle (that is, they lean to the right).
<code>otmEMSquare</code>	Specifies the number of logical units defining the x- or y-dimension of the em square for this font. (The number of units in the x- and y-directions are always the same for an em square.)
<code>otmAscent</code>	Specifies the maximum distance characters in this font extend above the base line. This is the typographic ascent for the font.
<code>otmDescent</code>	Specifies the maximum distance characters in this font extend below the base line. This is the typographic descent for the font.
<code>otmLineGap</code>	Specifies typographic line spacing.
<code>otmsXHeight</code>	Not supported.
<code>otmsCapEmHeight</code>	Not supported.
<code>otmrcFontBox</code>	Specifies the bounding box for the font.
<code>otmMacAscent</code>	Specifies the maximum distance characters in this font extend above the base line for the Macintosh.
<code>otmMacDescent</code>	Specifies the maximum distance characters in this font extend below the base line for the Macintosh.
<code>otmMacLineGap</code>	Specifies line-spacing information for the Macintosh.
<code>otmusMinimumPPEM</code>	Specifies the smallest recommended size for this font, in pixels per em-square.
<code>otmptSubscriptSize</code>	Specifies the recommended horizontal and vertical size for subscripts in this font.
<code>otmptSubscriptOffset</code>	Specifies the recommended horizontal and vertical offset for subscripts in this font. The subscript offset is measured from the character origin to the origin of the subscript character.
<code>otmptSuperscriptSize</code>	Specifies the recommended horizontal and vertical size for superscripts in this font.
<code>otmptSuperscriptOffset</code>	Specifies the recommended horizontal and vertical offset for superscripts in this font. The subscript offset is measured from the character base line to the base line of the superscript character.
<code>otmsStrikeoutSize</code>	Specifies the width of the strikeout stroke for this font. Typically, this is the width of the em-dash for the font.
<code>otmsStrikeoutPosition</code>	Specifies the position of the strikeout stroke relative to the base line for this font. Positive values are above the base line and negative values are below.
<code>otmsUnderscorePosition</code>	Specifies the position of the underscore character for this font.
<code>otmsUnderscoreSize</code>	Specifies the thickness of the underscore character for this font.
<code>otmpFamilyName</code>	Specifies the offset from the beginning of the structure to a string specifying the family name for the font.
<code>otmpFaceName</code>	Specifies the offset from the beginning of the structure to a string specifying the face name for the font. (This face name corresponds to the name specified in the <code>LOGFONT</code> structure.)
<code>otmpStyleName</code>	Specifies the offset from the beginning of the structure to a string specifying the style name for the font.
<code>otmpFullName</code>	Specifies the offset from the beginning of the structure to a string specifying the full name for the font. This name is unique for the font and often contains a version number or other identifying information.

#### Comments

The sizes returned in `OUTLINETEXMETRIC` are given in logical units; that is, they depend on the current mapping mode of the specified display context.



See Also

GetOutlineTextMetrics, PANOSE, TEXTMETRIC

■ **PAINTSTRUCT (2.x)**

```
typedef struct tagPAINTSTRUCT { /* ps */
    HDC   hdc;
    BOOL  fErase;
    RECT  rcPaint;
    BOOL  fRestore;
    BOOL  fIncUpdate;
    BYTE  rgbReserved[16];
} PAINTSTRUCT;
```

The PAINTSTRUCT structure contains information for an application. This information can be used to paint the client area of a window owned by that application.

Member	Description
hdc	Identifies the display context to be used for painting.
fErase	Specifies whether the background needs to be redrawn. This value is nonzero if the application should redraw the background. The application is responsible for drawing the background if a window class is created without a background brush. For more information, see the description of the hbrBackground member of the <u>WNDCLASS</u> structure.
rcPaint	Specifies the upper-left and lower-right corners of the rectangle in which the painting is requested.
fRestore	Reserved; used internally by Windows.
fIncUpdate	Reserved; used internally by Windows.
rgbReserved	Reserved (reserved memory object used internally by Windows)

See Also

BeginPaint, WNDCLASS

## Corrections

The description of the `fErase` member was reversed. It is nonzero if the application should redraw the background.

## PALETTEENTRY (3.0)

```
typedef struct tagPALETTEENTRY {    /* pe */
    BYTE  peRed;
    BYTE  peGreen;
    BYTE  peBlue;
    BYTE  peFlags;
} PALETTEENTRY;
```

The PALETTEENTRY structure specifies the color and usage of an entry in a logical color palette. A logical palette is defined by a LOGPALETTE structure.

Member	Description								
peRed	Specifies the intensity of red for the palette entry color.								
peGreen	Specifies the intensity of green for the palette entry color.								
peBlue	Specifies the intensity of blue for the palette entry color.								
peFlags	Specifies how the palette entry is to be used. The peFlags member may be set to NULL or to one of the following values (specifying NULL informs Windows that the palette entry contains an <u>RGB</u> value and that it should be mapped normally):								
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>PC_EXPLICIT</td><td>Specifies that the low-order word of the logical palette entry designates a hardware palette index. This flag allows the application to show the contents of the palette for the display device.</td></tr><tr><td>PC_NOCOLLAPSE</td><td>Specifies that the color will be placed in an unused entry in the system palette instead of being matched to an existing color in the system palette. Once this color is in the system palette, colors in other logical palettes can be matched to this color. If there are no unused entries in the system palette, the color is matched normally.</td></tr><tr><td>PC_RESERVED</td><td>Specifies that the logical palette entry will be used for palette animation. Because the color will frequently change, using this flag prevents other windows from matching colors to this palette entry. If an unused system-palette entry is available, this color is placed in that entry. Otherwise, the color will not be available for animation.</td></tr></tbody></table>	Value	Meaning	PC_EXPLICIT	Specifies that the low-order word of the logical palette entry designates a hardware palette index. This flag allows the application to show the contents of the palette for the display device.	PC_NOCOLLAPSE	Specifies that the color will be placed in an unused entry in the system palette instead of being matched to an existing color in the system palette. Once this color is in the system palette, colors in other logical palettes can be matched to this color. If there are no unused entries in the system palette, the color is matched normally.	PC_RESERVED	Specifies that the logical palette entry will be used for palette animation. Because the color will frequently change, using this flag prevents other windows from matching colors to this palette entry. If an unused system-palette entry is available, this color is placed in that entry. Otherwise, the color will not be available for animation.
Value	Meaning								
PC_EXPLICIT	Specifies that the low-order word of the logical palette entry designates a hardware palette index. This flag allows the application to show the contents of the palette for the display device.								
PC_NOCOLLAPSE	Specifies that the color will be placed in an unused entry in the system palette instead of being matched to an existing color in the system palette. Once this color is in the system palette, colors in other logical palettes can be matched to this color. If there are no unused entries in the system palette, the color is matched normally.								
PC_RESERVED	Specifies that the logical palette entry will be used for palette animation. Because the color will frequently change, using this flag prevents other windows from matching colors to this palette entry. If an unused system-palette entry is available, this color is placed in that entry. Otherwise, the color will not be available for animation.								

See Also

AnimatePalette, LOGPALETTE

## PANOSE (3.1)

```
typedef struct tagPANOSE {      /* panose */
    BYTE bFamilyType;
    BYTE bSerifStyle;
    BYTE bWeight;
    BYTE bProportion;
    BYTE bContrast;
    BYTE bStrokeVariation;
    BYTE bArmStyle;
    BYTE bLetterform;
    BYTE bMidline;
    BYTE bXHeight;
} PANOSE;
```

The PANOSE structure describes the Panose font-classification values for a TrueType font.

Member	Description																																		
bFamilyType	Specifies the font family. This member can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Any</td></tr><tr><td>1</td><td>No fit</td></tr><tr><td>2</td><td>Text and display</td></tr><tr><td>3</td><td>Script</td></tr><tr><td>4</td><td>Decorative</td></tr><tr><td>5</td><td>Pictorial</td></tr></tbody></table>	Value	Meaning	0	Any	1	No fit	2	Text and display	3	Script	4	Decorative	5	Pictorial																				
Value	Meaning																																		
0	Any																																		
1	No fit																																		
2	Text and display																																		
3	Script																																		
4	Decorative																																		
5	Pictorial																																		
bSerifStyle	Specifies the style of serifs for the font. This member can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Any</td></tr><tr><td>1</td><td>No fit</td></tr><tr><td>2</td><td>Cove</td></tr><tr><td>3</td><td>Obtuse cove</td></tr><tr><td>4</td><td>Square cove</td></tr><tr><td>5</td><td>Obtuse square cove</td></tr><tr><td>6</td><td>Square</td></tr><tr><td>7</td><td>Thin</td></tr><tr><td>8</td><td>Bone</td></tr><tr><td>9</td><td>Exaggerated</td></tr><tr><td>10</td><td>Triangle</td></tr><tr><td>11</td><td>Normal sans</td></tr><tr><td>12</td><td>Obtuse sans</td></tr><tr><td>13</td><td>Perp sans</td></tr><tr><td>14</td><td>Flared</td></tr><tr><td>15</td><td>Rounded</td></tr></tbody></table>	Value	Meaning	0	Any	1	No fit	2	Cove	3	Obtuse cove	4	Square cove	5	Obtuse square cove	6	Square	7	Thin	8	Bone	9	Exaggerated	10	Triangle	11	Normal sans	12	Obtuse sans	13	Perp sans	14	Flared	15	Rounded
Value	Meaning																																		
0	Any																																		
1	No fit																																		
2	Cove																																		
3	Obtuse cove																																		
4	Square cove																																		
5	Obtuse square cove																																		
6	Square																																		
7	Thin																																		
8	Bone																																		
9	Exaggerated																																		
10	Triangle																																		
11	Normal sans																																		
12	Obtuse sans																																		
13	Perp sans																																		
14	Flared																																		
15	Rounded																																		
bWeight	Specifies the weight of the font. This member can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Any</td></tr><tr><td>1</td><td>No fit</td></tr><tr><td>2</td><td>Very light</td></tr><tr><td>3</td><td>Light</td></tr><tr><td>4</td><td>Thin</td></tr><tr><td>5</td><td>Book</td></tr><tr><td>6</td><td>Medium</td></tr><tr><td>7</td><td>Demi</td></tr></tbody></table>	Value	Meaning	0	Any	1	No fit	2	Very light	3	Light	4	Thin	5	Book	6	Medium	7	Demi																
Value	Meaning																																		
0	Any																																		
1	No fit																																		
2	Very light																																		
3	Light																																		
4	Thin																																		
5	Book																																		
6	Medium																																		
7	Demi																																		

	8	Bold
	9	Heavy
	10	Black
	11	Nord
<b>bProportion</b>	Specifies the proportion of the font. This member can be one of the following values:	
	Value	Meaning
	0	Any
	1	No fit
	2	Old style
	3	Modern
	4	Even width
	5	Expanded
	6	Condensed
	7	Very expanded
	8	Very condensed
	9	Monospaced
<b>bContrast</b>	Specifies the contrast of the font. This member can be one of the following values:	
	Value	Meaning
	0	Any
	1	No fit
	2	None
	3	Very low
	4	Low
	5	Medium low
	6	Medium
	7	Medium high
	8	High
	9	Very high
<b>bStrokeVariation</b>	Specifies the stroke variation for the font. This member can be one of the following values:	
	Value	Meaning
	0	Any
	1	No fit
	2	Gradual/diagonal
	3	Gradual/transitional
	4	Gradual/vertical
	5	Gradual/horizontal
	6	Rapid/vertical
	7	Rapid/horizontal
	8	Instant/vertical
<b>bArmStyle</b>	Specifies the style for the arms in the font. This member can be one of the following values:	
	Value	Meaning
	0	Any
	1	No fit
	2	Straight arms/horizontal
	3	Straight arms/wedge
	4	Straight arms/vertical
	5	Straight arms/single serif
	6	Straight arms/double serif

	7	Non-straight arms/horizontal
	8	Non-straight arms/wedge
	9	Non-straight arms/vertical
	10	Non-straight arms/single serif
	11	Non-straight arms/double serif
<b>bLetterform</b>	Specifies the letter form for the font. This member can be one of the following values:	
	<b>Value</b>	<b>Meaning</b>
	0	Any
	1	No fit
	2	Normal/contact
	3	Normal/weighted
	4	Normal/boxed
	5	Normal/flattened
	6	Normal/rounded
	7	Normal/off-center
	8	Normal/square
	9	Oblique/contact
	10	Oblique/weighted
	11	Oblique/boxed
	12	Oblique/flattened
	13	Oblique/rounded
	14	Oblique/off-center
	15	Oblique/square
<b>bMidline</b>	Specifies the style of the midline for the font. This member can be one of the following values:	
	<b>Value</b>	<b>Meaning</b>
	0	Any
	1	No fit
	2	Standard/trimmed
	3	Standard/pointed
	4	Standard/serifed
	5	High/trimmed
	6	High/pointed
	7	High/serifed
	8	Constant/trimmed
	9	Constant/pointed
	10	Constant/serifed
	11	Low/trimmed
	12	Low/pointed
	13	Low/serifed
<b>bXHeight</b>	Specifies the X-height of the font. This member can be one of the following values:	
	<b>Value</b>	<b>Meaning</b>
	0	Any
	1	No fit
	2	Constant/small
	3	Constant/standard
	4	Constant/large
	5	Ducking/small
	6	Ducking/standard
	7	Ducking/large

See Also

OUTLINETEXTMETRIC



## POINT (2.x)

```
typedef struct tagPOINT {    /* pt */
    int x;
    int y;
} POINT;
```

The POINT structure defines the x- and y-coordinates of a point.

Member	Description
x	Specifies the x-coordinate of a point.
y	Specifies the y-coordinate of a point.

See Also

[ChildWindowFromPoint](#), [PtInRect](#), [WindowFromPoint](#)

## POINTFX (3.1)

```
typedef struct tagPOINTFX {  
    FIXED x;  
    FIXED y;  
} POINTFX;
```

The POINTFX structure contains the coordinates of points that describe the outline of a character in a TrueType font. POINTFX is a member of the TTPOLYCURVE and TTPOLYGONHEADER structures.

Member	Description
x	Specifies the x-component of a point on the outline of a TrueType character.
y	Specifies the y-component of a point on the outline of a TrueType character.

See Also

FIXED, TTPOLYCURVE, TTPOLYGONHEADER, GetGlyphOutline

## PRINTDLG (3.1)

```
#include <commdlg.h>

typedef struct tagPD { /* pd */
    DWORD lStructSize;
    HWND hwndOwner;
    HGLOBAL hDevMode;
    HGLOBAL hDevNames;
    HDC hDC;
    DWORD Flags;
    UINT nFromPage;
    UINT nToPage;
    UINT nMinPage;
    UINT nMaxPage;
    UINT nCopies;
    HINSTANCE hInstance;
    LPARAM lCustData;
    UINT (CALLBACK* lpfnPrintHook)(HWND, UINT, WPARAM, LPARAM);
    UINT (CALLBACK* lpfnSetupHook)(HWND, UINT, WPARAM, LPARAM);
    LPCSTR lpPrintTemplateName;
    LPCSTR lpSetupTemplateName;
    HGLOBAL hPrintTemplate;
    HGLOBAL hSetupTemplate;
} PRINTDLG;
```

The PRINTDLG structure contains information that the system uses to initialize the system-defined Print dialog box. After the user chooses the OK button to close the dialog box, the system returns information about the user's selections in this structure.

Member	Description
lStructSize	Specifies the length of the structure, in bytes. This member is filled on input.
hwndOwner	Identifies the window that owns the dialog box. This member can be any valid window handle, or it should be NULL if the dialog box is to have no owner.  If the PD_SHOWHELP flag is set, hwndOwner must identify the window that owns the dialog box. The window procedure for this owner window receives a notification message when the user chooses the Help button. (The identifier for the notification message is the value returned by the RegisterWindowMessage function when HELPPMSGSTRING is passed as its argument.)
hDevMode	This member is filled on input.  Identifies a movable global memory object that contains a <u>DEVMODE</u> structure. Before the <u>PrintDlg</u> function is called, the members in this structure may contain <u>data</u> used to initialize the dialog box controls. When the <u>PrintDlg</u> function returns, the members in this structure specify the state of each of the dialog box controls.  If the application uses the structure to initialize the dialog box controls, it must allocate space for and create the <u>DEVMODE</u> structure. (The application should allocate a movable memory object.)  If the application does not use the structure to initialize the dialog box controls, the hDevMode member may be NULL. In this case, the <u>PrintDlg</u> function allocates memory for the structure, initializes its members, and returns a handle that identifies it.  If the device driver for the specified printer does not support extended device modes, the hDevMode member is NULL when <u>PrintDlg</u> returns.  If the device name (specified by the dmDeviceName member of the <u>DEVMODE</u> structure) does not appear in the [devices] section of WIN.INI, the <u>PrintDlg</u> function returns an error.  The value of hDevMode may change during the execution of the <u>PrintDlg</u> function. If this value changes, the <u>PrintDlg</u> function has

already freed the original handle and allocated a new one. When the calling application is finished with the handle, it must free it by calling the `GlobalFree` function. This value may change even if the `PrintDlg` function returns zero.

This member is filled on input and output.

`hDevNames`

Identifies a movable global memory object that contains a `DEVNAMES` structure. This structure contains three strings; these strings specify the driver name, the printer name, and the output-port name. Before the `PrintDlg` function is called, the members of this structure contain strings used to initialize the dialog box controls. When the `PrintDlg` function returns, the members of this structure contain the strings typed by the user. The calling application uses these strings to create a device context or an information context.

If the application uses the structure to initialize the dialog box controls, it must allocate space for and create the `DEVMODE` data structure. (The application should allocate a movable global memory object.)

If the application does not use the structure to initialize the dialog box controls, the `hDevNames` member can be `NULL`. In this case, the `PrintDlg` function allocates memory for the structure, initializes its members (using the printer name specified in the `DEVMODE` data structure), and returns a handle that identifies it. When the `PrintDlg` function initializes the members of the `DEVNAMES` structure, it uses the first port name that appears in the `[devices]` section of `WIN.INI`. For example, the function uses "LPT1" as the port name if the following string appears in the `[devices]` section:

```
PCL / HP LaserJet=HPPCL,LPT1:,LPT2:
```

If both the `hDevMode` and `hDevNames` members are `NULL`, `PrintDlg` specifies the current default printer for `hDevNames`.

The value of `hDevNames` may change during the execution of the `PrintDlg` function. If this value changes, the `PrintDlg` function has already freed the original handle and allocated a new one. When the calling application is finished with the handle, it must free it by calling the `GlobalFree` function. This value may change even if the `PrintDlg` function returns zero.

This member is filled on input and output.

`hDC`

Identifies either a device context or an information context, depending on whether the `Flags` member specifies the `PD_RETURNDC` or the `PC_RETURNIC` flag. If neither flag is specified, the value of this member is undefined. If both flags are specified, `hDC` is `PD_RETURNDC`.

This member is filled on output.

`Flags`

Specifies the dialog box initialization flags. This member may be a combination of the following values:

Value	Meaning
<code>PD_ALLPAGES</code>	Indicates that the All radio button was selected when the user closed the dialog box. (This value is used as a placeholder, to indicate that the <code>PD_PAGENUMS</code> and <code>PD_SELECTION</code> flags are not set. This value

PD\_COLLATE

can be set on input and output.)

Causes the Collate Copies check box to be checked when the dialog box is created. When the PrintDlg function returns, this flag indicates the state in which the user left the Collate Copies check box. This flag can be set on input and output.

PD\_DISABLEPRINTTOFILE

Disables the Print to File check box.

PD\_ENABLEPRINTHOOK

Enables the hook function specified in the lpfnPrintHook member of this structure.

PD\_ENABLEPRINTTEMPLATE

Causes the system to use the dialog box template identified by the hInstance and lpPrintTemplateName members to create the Print dialog box.

PD\_ENABLEPRINTTEMPLATEHANDLE

Indicates that the hPrintTemplate member identifies a data block that contains a pre-loaded dialog box template. The system ignores the hInstance member if this flag is specified.

PD\_ENABLESETUPHOOK

Enables the hook function specified in the lpfnSetupHook member of this structure.

PD_ENABLESETUPTEMPLATE	Causes the system to use the dialog box template identified by the hInstance and lpSetupTemplateName members to create the Print Setup dialog box.
PD_ENABLESETUPTEMPLATEHANDLE	Indicates that the hSetupTemplate member identifies a data block that contains a pre-loaded dialog box template. The system ignores the hInstance member if this flag is specified.
PD_HIDEPRINTTOFILE	Hides and disables the Print to File check box.
PD_NOPAGENUMS	Disables the Pages radio button and the associated edit controls.
PD_NOSELECTION	Disables the Selection radio button.
PD_NOWARNING	Prevents the warning message from being displayed when there is no default printer.
PD_PAGENUMS	Causes the Pages radio button to be selected when the dialog box is created. When the <u>PrintDlg</u> function returns, this flag is set if the Pages button is in the selected state. If neither PD_PAGENUMS nor

	<p>PD_SELECTION is specified, the All radio button is in the selected state.</p> <p>This flag can be set on input and output.</p>
PD_PRINTSETUP	<p>Causes the system to display the Print Setup dialog box rather than the Print dialog box.</p>
PD_PRINTTOFILE	<p>Causes the Print to File check box to be checked when the dialog box is created.</p> <p>This flag can be set on input and output.</p>
PD_RETURNDC	<p>Causes the <u>PrintDlg</u> function to return a device context matching the selections that the user made in the dialog box. The handle to the device context is returned in the hDC member. If neither PD_RETURNDC nor PD_RETURNIC is specified, the hDC parameter is undefined on output.</p>
PD_RETURNDEFAULT	<p>Causes the <u>PrintDlg</u> function to return <u>DEVMODE</u> and <u>DEVNAMES</u> structures that are initialized for the system default printer. PrintDlg does this without</p>

displaying a dialog box. Both the hDevNames and the hDevMode members should be NULL; otherwise, the function returns an error. If the system default printer is supported by an old printer driver (earlier than Windows version 3.0), only the hDevNames member is returned--the hDevMode member is NULL.

PD\_RETURNIC

Causes the PrintDlg function to return an information context matching the selections that the user made in the dialog box. The information context is returned in the hDC member. If neither PD\_RETURNDC nor PD\_RETURNIC is specified, the hDC parameter is undefined on output.

PD\_SELECTION

Causes the Selection radio button to be selected when the dialog box is created. When the PrintDlg function returns, this flag is set if the Selection button is in the



selected state.  
If neither  
PD\_PAGENUMS  
nor  
PD\_SELECTION  
is specified,  
the All radio  
button is in the  
selected state.

This flag can  
be set on input  
and output.

PD\_SHOWHELP

Causes the  
dialog box to  
show the Help  
button. If this  
flag is  
specified, the  
hwndOwner  
must not be  
NULL.

PD\_USEDEVMODECOPIES

Disables the  
Copies edit  
control if a  
printer driver  
does not  
support  
multiple  
copies. If a  
driver does  
support  
multiple  
copies, setting  
this flag  
indicates that  
the PrintDlg  
function  
should store  
the requested  
number of  
copies in the  
dmCopies  
member of the  
DEVMODE  
structure and  
store the value  
1 in the  
nCopies  
member of the  
PRINTDLG  
structure.

If this flag is  
not set, the  
PRINTDLG  
structure stores  
the value 1 in  
the dmCopies  
member of the  
DEVMODE  
structure and  
stores the  
requested  
number of

copies in the  
nCopies  
member of the  
PRINTDLG  
structure.

	<p>These flags may be set when the structure is initialized, except where specified.</p>
nFromPage	<p>Specifies the initial value for the starting page in the From edit control. When the <u>PrintDlg</u> function returns, this member specifies the page at which to begin printing. This value is valid only if the PD_PAGENUMS flag is specified. The maximum value for this member is 0xFFFFE; if 0xFFFF is specified, the From edit control is left blank.</p>
nToPage	<p>This member is filled on input and output.</p> <p>Specifies the initial value for the ending page in the To edit control. When the <u>PrintDlg</u> function returns, this member specifies the last page to print. This value is valid only if the PD_PAGENUMS flag is specified. The maximum value for this member is 0xFFFFE; if 0xFFFF is specified, the To edit control is left blank.</p>
nMinPage	<p>This member is filled on input and output.</p> <p>Specifies the minimum number of pages that can be specified in the From and To edit controls. This member is filled on input.</p>
nMaxPage	<p>Specifies the maximum number of pages that can be specified in the From and To edit controls. This member is filled on input.</p>
nCopies	<p>Before the <u>PrintDlg</u> function is called, this member specifies the value to be used to initialize the Copies edit control if the hDevMode member is NULL; otherwise, the dmCopies member of the <u>DEVMODE</u> structure contains the value used to initialize the Copies edit control.</p> <p>When <u>PrintDlg</u> returns, the value specified by this member depends on the version of Windows for which the printer driver was written. For printer drivers written for Windows versions earlier than 3.0, this member specifies the number of copies requested by the user in the Copies edit control. For printer drivers written for Windows versions 3.0 and later, this member specifies the number of copies requested by the user if the PD_USEDEVMODECOPIES flag was not set; otherwise, this member specifies the value 1 and the actual number of copies requested appears in the <u>DEVMODE</u> structure.</p>
hInstance	<p>This member is filled on input and output.</p> <p>Identifies a data block that contains the pre-loaded dialog box template specified by the lpPrintTemplateName or the lpSetupTemplateName member. This member is used only if the Flags member specifies the PD_ENABLEPRINTTEMPLATE or PD_ENABLESETUPTEMPLATE flag; otherwise, this member is ignored.</p>
lCustData	<p>This member is filled on input.</p> <p>Specifies application-defined data that the system passes to the hook function identified by the lpfnPrintHook or the lpfnSetupHook member. The system passes a pointer to the PRINTDLG structure in the lParam parameter of the <u>WM_INITDIALOG</u> message; this pointer can be used to retrieve the lCustData member.</p>
lpfnPrintHook	<p>Points to the exported hook function that processes dialog box messages if the application customizes the Print dialog box. This member is ignored unless the PD_ENABLEPRINTHOOK flag is specified in the Flags member.</p>
lpfnSetupHook	<p>This member is filled on input.</p> <p>Points to the exported hook function that processes dialog box messages if the application customizes the Print Setup dialog box. This member is ignored unless the PD_ENABLESETUPHOOK flag is specified in the Flags member.</p> <p>This member is filled on input.</p>

lpPrintTemplateName	Points to a null-terminated string that specifies the dialog box template that is to be substituted for the standard dialog box template in COMMDLG. An application must specify the PD_ENABLEPRINTTEMPLATE constant in the Flags member to enable the hook function; otherwise, the system ignores this structure member. This member is filled on input.
lpSetupTemplateName	Points to a null-terminated string that specifies the dialog box template that is to be substituted for the standard dialog box template in COMMDLG. An application must specify the PD_ENABLEPRINTTEMPLATE constant in the Flags member to enable the hook function; otherwise, the system ignores this structure member. This member is filled on input.
hPrintTemplate	Identifies the handle of the global memory object that contains the pre-loaded dialog box template to be used instead of the default template in COMMDLG.DLL for the Print dialog box. To use the dialog box template, the PD_ENABLEPRINTTEMPLATEHANDLE flag must be set. This member is filled on input.
hSetupTemplate	Identifies the handle of the global memory object that contains the pre-loaded dialog box template to be used instead of the default template in COMMDLG.DLL for the Print Setup dialog box. To use the dialog box template, the PD_ENABLEPRINTTEMPLATEHANDLE flag must be set. This member is filled on input.

See Also

CreateDC, CreateIC, PrintDlg, DEVMODE, DEVNAMES

## RASTERIZER\_STATUS (3.1)

```
typedef struct tagRASTERIZER_STATUS {    /* rs */
    int    nSize;
    int    wFlags;
    int    nLanguageID;
} RASTERIZER_STATUS;
```

The RASTERIZER\_STATUS structure contains information about whether TrueType is installed. This structure is filled when an application calls the GetRasterizerCaps function.

Member	Description
nSize	Specifies the size, in bytes, of the RASTERIZER_STATUS structure.
wFlags	Specifies whether at least one TrueType font is installed and whether TrueType is enabled. This value is TT_AVAILABLE and/or TT_ENABLED if TrueType is on the system.
nLanguageID	Specifies the language in the system's SETUP.INF file.

See Also  
GetRasterizerCaps

## RECT (2.x)

```
typedef struct tagRECT {    /* rc */
    int left;
    int top;
    int right;
    int bottom;
} RECT;
```

The RECT structure defines the coordinates of the upper-left and lower-right corners of a rectangle.

Member	Description
left	Specifies the x-coordinate of the upper-left corner of a rectangle.
top	Specifies the y-coordinate of the upper-left corner of a rectangle.
right	Specifies the x-coordinate of the lower-right corner of a rectangle.
bottom	Specifies the y-coordinate of the lower-right corner of a rectangle.

### Comments

The width of the rectangle defined by the RECT structure must not exceed 32,767 units.

When the RECT structure is passed to the FillRect function, graphics device interface (GDI) fills the rectangle up to, but not including, the right column and bottom row of pixels.

## RGBQUAD (3.0)

```
typedef struct tagRGBQUAD { /* rgbq */
    BYTE    rgbBlue;
    BYTE    rgbGreen;
    BYTE    rgbRed;
    BYTE    rgbReserved;
} RGBQUAD;
```

The RGBQUAD structure describes a color consisting of relative intensities of red, green, and blue. The bmiColors member of the BITMAPINFO structure consists of an array of RGBQUAD structures.

Member	Description
rgbBlue	Specifies the intensity of blue in the color.
rgbGreen	Specifies the intensity of green in the color.
rgbRed	Specifies the intensity of red in the color.
rgbReserved	Not used; must be set to zero.

See Also

BITMAPINFO

## RGBTRIPLE (3.0)

```
typedef struct tagRGBTRIPLE {    /* rgbt */
    BYTE    rgbtBlue;
    BYTE    rgbtGreen;
    BYTE    rgbtRed;
} RGBTRIPLE;
```

The RGBTRIPLE structure describes a color consisting of relative intensities of red, green, and blue. The `bmciColors` member of the BITMAPCOREINFO structure consists of an array of RGBTRIPLE structures.

Windows applications should use the BITMAPINFO structure instead of BITMAPCOREINFO whenever possible. The BITMAPINFO structure uses an RGBQUAD structure instead of the RGBTRIPLE structure.

Member	Description
<code>rgbtBlue</code>	Specifies the intensity of blue in the color.
<code>rgbtGreen</code>	Specifies the intensity of green in the color.
<code>rgbtRed</code>	Specifies the intensity of red in the color.

See Also

BITMAPCOREINFO, BITMAPINFO, RGBQUAD

## SEGINFO (3.1)

```
typedef struct tagSEGINFO {    /* segi */
    UINT    offSegment;
    UINT    cbSegment;
    UINT    flags;
    UINT    cbAlloc;
    HGLOBAL h;
    UINT    alignShift;
    UINT    reserved[2];
} SEGINFO;
```

The SEGINFO structure contains information about a data or code segment. This structure is filled in by the [GetCodeInfo](#) function.

Member	Description																		
offSegment	Specifies the offset, in sectors, to the contents of the segment data, relative to the beginning of the file. (Zero means no file data is available.) The size of the sector is determined by shifting left by 1 the value given in the alignShift member.																		
cbSegment	Specifies the length of the segment in the file, in bytes. Zero means 64K.																		
flags	Contains flags which specify attributes of the segment. The following list describes these flags: <table><thead><tr><th>Bit</th><th>Meaning</th></tr></thead><tbody><tr><td>0-2</td><td>Specifies the segment type. If bit 0 is set to 1, the segment is a data segment. Otherwise, the segment is a code segment.</td></tr><tr><td>3</td><td>Specifies whether segment data is iterated. When this bit is set to 1, the segment data is iterated.</td></tr><tr><td>4</td><td>Specifies whether the segment is movable or fixed. When this bit is set to 1, the segment is movable. Otherwise, it is fixed.</td></tr><tr><td>5-6</td><td>Reserved.</td></tr><tr><td>7</td><td>Specifies whether the segment is a read-only data segment or an execute-only code segment. If this bit is set to 1 and the segment is a code segment, the segment is an execute-only segment. If this bit is set to zero and the segment is a data segment, it is a read-only segment.</td></tr><tr><td>8</td><td>Specifies whether the segment has associated relocation information. If this bit is set to 1, the segment has relocation information. Otherwise, the segment does not have relocation information.</td></tr><tr><td>9</td><td>Specifies whether the segment has debugging information. If this bit is set to 1, the segment has debugging information. Otherwise, the segment does not have debugging information.</td></tr><tr><td>10-15</td><td>Reserved.</td></tr></tbody></table>	Bit	Meaning	0-2	Specifies the segment type. If bit 0 is set to 1, the segment is a data segment. Otherwise, the segment is a code segment.	3	Specifies whether segment data is iterated. When this bit is set to 1, the segment data is iterated.	4	Specifies whether the segment is movable or fixed. When this bit is set to 1, the segment is movable. Otherwise, it is fixed.	5-6	Reserved.	7	Specifies whether the segment is a read-only data segment or an execute-only code segment. If this bit is set to 1 and the segment is a code segment, the segment is an execute-only segment. If this bit is set to zero and the segment is a data segment, it is a read-only segment.	8	Specifies whether the segment has associated relocation information. If this bit is set to 1, the segment has relocation information. Otherwise, the segment does not have relocation information.	9	Specifies whether the segment has debugging information. If this bit is set to 1, the segment has debugging information. Otherwise, the segment does not have debugging information.	10-15	Reserved.
Bit	Meaning																		
0-2	Specifies the segment type. If bit 0 is set to 1, the segment is a data segment. Otherwise, the segment is a code segment.																		
3	Specifies whether segment data is iterated. When this bit is set to 1, the segment data is iterated.																		
4	Specifies whether the segment is movable or fixed. When this bit is set to 1, the segment is movable. Otherwise, it is fixed.																		
5-6	Reserved.																		
7	Specifies whether the segment is a read-only data segment or an execute-only code segment. If this bit is set to 1 and the segment is a code segment, the segment is an execute-only segment. If this bit is set to zero and the segment is a data segment, it is a read-only segment.																		
8	Specifies whether the segment has associated relocation information. If this bit is set to 1, the segment has relocation information. Otherwise, the segment does not have relocation information.																		
9	Specifies whether the segment has debugging information. If this bit is set to 1, the segment has debugging information. Otherwise, the segment does not have debugging information.																		
10-15	Reserved.																		
cbAlloc	Specifies the total amount of memory allocated for the segment. This amount may exceed the actual size of the segment. Zero means 64K.																		
h	Identifies the global memory for the segment.																		
alignShift	Specifies the size of the addressable sector as an exponent of 2. An executable file pads the application's code, data, and resource segments with zero bytes so that the segments are always a multiple of the file-segment size. Windows discards the extra bytes when it loads the segments from the file.																		
reserved	Specifies two reserved <a href="#">UINT</a> values.																		

See Also  
[GetCodeInfo](#)



## SIZE (3.1)

```
typedef struct tagSIZE {  
    int cx;  
    int cy;  
} SIZE;
```

The `SIZE` structure contains viewport extents, window extents, text extents, bitmap dimensions, and the aspect-ratio filter for some extended functions for Windows 3.1

Member	Description
<code>cx</code>	Specifies the x-extent when a function returns.
<code>cy</code>	Specifies the y-extent when a function returns.

### See Also

[GetAspectRatioFilterEx](#), [GetBitmapDimensionEx](#), [GetTextExtentPoint](#), [GetViewportExtEx](#), [GetWindowExtEx](#), [ScaleViewportExtEx](#), [ScaleWindowExtEx](#), [SetBitmapDimensionEx](#), [SetViewportExtEx](#), [SetWindowExtEx](#)

## STACKTRACEENTRY (3.1)

```
#include <toolhelp.h>
```

```
typedef struct tagSTACKTRACEENTRY { /* ste */
    DWORD    dwSize;
    HTASK    hTask;
    WORD     wSS;
    WORD     wBP;
    WORD     wCS;
    WORD     wIP;
    HMODULE  hModule;
    WORD     wSegment;
    WORD     wFlags;
} STACKTRACEENTRY;
```

The STACKTRACEENTRY structure contains information about one stack frame. This information enables an application to trace back through the stack of a specific task.

Member	Description						
dwSize	Specifies the size of the STACKTRACEENTRY structure, in bytes.						
hTask	Identifies the task handle for the stack.						
wSS	Contains the value in the SS register. This value is used with the value of the wBP member to determine the next entry in the stack-trace table.						
wBP	Contains the value in the BP register. This value is used with the wSS value to determine the next entry in the stack-trace table.						
wCS	Contains the value in the CS register on return. This value is used with the value of the wIP member to determine the return value of the function.						
wIP	Contains the value in the IP register on return. This value is used with the wCS value to determine the return value of the function.						
hModule	Identifies the module that contains the currently executing function.						
wSegment	Contains the segment number of the current selector.						
wFlags	Indicates the frame type. This type can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>FRAME_FAR</td><td>The CS register contains a valid code segment.</td></tr><tr><td>FRAME_NEAR</td><td>The CS register is null.</td></tr></tbody></table>	Value	Meaning	FRAME_FAR	The CS register contains a valid code segment.	FRAME_NEAR	The CS register is null.
Value	Meaning						
FRAME_FAR	The CS register contains a valid code segment.						
FRAME_NEAR	The CS register is null.						

See Also

[StackTraceCSIPFirst](#), [StackTraceNext](#), [StackTraceFirst](#)

## SYSHEAPINFO (3.1)

```
#include <toolhelp.h>

typedef struct tagSYSHEAPINFO { /* shi */
    DWORD    dwSize;
    WORD     wUserFreePercent;
    WORD     wGDIFreePercent;
    HGLOBAL  hUserSegment;
    HGLOBAL  hGDIsegment;
} SYSHEAPINFO;
```

The SYSHEAPINFO structure contains information about the USER and GDI modules.

Member	Description
dwSize	Specifies the size of the SYSHEAPINFO structure, in bytes.
wUserFreePercent	Specifies the percentage of the USER local heap that is free.
wGDIFreePercent	Specifies the percentage of the GDI local heap that is free.
hUserSegment	Identifies the DGROUP segment of the USER local heap.
hGDIsegment	Identifies the DGROUP segment of the GDI local heap.

See Also

[SystemHeapInfo](#)

## TASKENTRY (3.1)

```
#include <toolhelp.h>

typedef struct tagTASKENTRY { /* te */
    DWORD dwSize;
    HTASK hTask;
    HTASK hTaskParent;
    HINSTANCE hInst;
    HMODULE hModule;
    WORD wSS;
    WORD wSP;
    WORD wStackTop;
    WORD wStackMinimum;
    WORD wStackBottom;
    WORD wcEvents;
    HGLOBAL hQueue;
    char szModule[MAX_MODULE_NAME + 1];
    WORD wPSPOffset;
    HANDLE hNext;
} TASKENTRY;
```

The TASKENTRY structure contains information about one task.

Member	Description
dwSize	Specifies the size of the TASKENTRY structure, in bytes.
hTask	Identifies the task handle for the stack.
hTaskParent	Identifies the parent of the task.
hInst	Identifies the instance handle of the task. This value is equivalent to the task's DGROUP segment selector.
hModule	Identifies the module that contains the currently executing function.
wSS	Contains the value in the SS register.
wSP	Contains the value in the SP register.
wStackTop	Specifies the offset to the top of the stack (lowest address on the stack).
wStackMinimum	Specifies the lowest segment number of the stack during execution of the task.
wStackBottom	Specifies the offset to the bottom of the stack (highest address on the stack).
wcEvents	Specifies the number of pending events.
hQueue	Identifies the task queue.
szModule	Specifies the name of the module that contains the currently executing function.
wPSPOffset	Specifies the offset from the program segment prefix (PSP) to the beginning of the executable code segment.
hNext	Identifies the next entry in the task list. This member is reserved for internal use by Windows.

See Also

[TaskFindHandle](#), [TaskFirst](#), [TaskNext](#)

## TEXTMETRIC (2.x)

```
typedef struct tagTEXTMETRIC { /* tm */
    int tmHeight;
    int tmAscent;
    int tmDescent;
    int tmInternalLeading;
    int tmExternalLeading;
    int tmAveCharWidth;
    int tmMaxCharWidth;
    int tmWeight;
    BYTE tmItalic;
    BYTE tmUnderlined;
    BYTE tmStruckOut;
    BYTE tmFirstChar;
    BYTE tmLastChar;
    BYTE tmDefaultChar;
    BYTE tmBreakChar;
    BYTE tmPitchAndFamily;
    BYTE tmCharSet;
    int tmOverhang;
    int tmDigitizedAspectX;
    int tmDigitizedAspectY;
} TEXTMETRIC;
```

The TEXTMETRIC structure contains basic information about a physical font. For Windows version 3.1 and later, the [EnumFonts](#) and [EnumFontFamilies](#) functions return information about TrueType fonts in a [NEWTEXTMETRIC](#) structure.

Member	Description								
tmHeight	Specifies the height of character cells. (The height is the sum of the tmAscent and tmDescent members.)								
tmAscent	Specifies the ascent of character cells. (The ascent is the space between the base line and the top of the character cell.)								
tmDescent	Specifies the descent of character cells. (The descent is the space between the bottom of the character cell and the base line.)								
tmInternalLeading	Specifies the difference between the point size of a font and the physical size of the font. For TrueType fonts, this value is equal to tmHeight minus (s * ntmSizeEM), where s is the scaling factor for the TrueType font and ntmSizeEM is a value from the <a href="#">NEWTEXTMETRIC</a> structure. For bitmap fonts, this value is used to determine the point size of a font. When an application specifies a negative value in the lfHeight member of the <a href="#">LOGFONT</a> structure, the application is requesting a font whose height equals tmHeight minus tmInternalLeading.								
tmExternalLeading	Specifies the amount of extra leading (space) that the application adds between rows. Since this area is outside the character cell, it contains no marks and will not be altered by text output calls in either opaque or transparent mode. The font designer sometimes sets this member to zero.								
tmAveCharWidth	Specifies the average width of characters in the font. For ANSI_CHARSET fonts, this is a weighted average of the characters "a" through "z" and the space character. For other character sets, this value is an unweighted average of all characters in the font.								
tmMaxCharWidth	Specifies the "B" spacing of the widest character in the font. For more information about "B" spacing, see the description of the <a href="#">ABC</a> structure.								
tmWeight	Specifies the weight of the font. This member can be one of the following values: <table><thead><tr><th>Constant</th><th>Value</th></tr></thead><tbody><tr><td>FW_DONTCARE</td><td>0</td></tr><tr><td>FW_THIN</td><td>100</td></tr><tr><td>FW_EXTRALIGHT</td><td>200</td></tr></tbody></table>	Constant	Value	FW_DONTCARE	0	FW_THIN	100	FW_EXTRALIGHT	200
Constant	Value								
FW_DONTCARE	0								
FW_THIN	100								
FW_EXTRALIGHT	200								

FW_ULTRALIGHT	200
FW_LIGHT	300
FW_NORMAL	400
FW_REGULAR	400
FW_MEDIUM	500
FW_SEMIBOLD	600
FW_DEMIBOLD	600
FW_BOLD	700
FW_EXTRABOLD	800
FW_ULTRABOLD	800
FW_BLACK	900
FW_HEAVY	900

tmItalic	Specifies an italic font if it is nonzero.
tmUnderlined	Specifies an underlined font if it is nonzero.
tmStruckOut	Specifies a "struckout" font if it is nonzero.
tmFirstChar	Specifies the value of the first character defined in the font.
tmLastChar	Specifies the value of the last character defined in the font.
tmDefaultChar	Specifies the value of the character that will be substituted for characters that are not in the font.
tmBreakChar	Specifies the value of the character that will be used to define word breaks for text justification.
tmPitchAndFamily	Specifies the pitch and family of the selected font.

The four low-order bits identify the type of font, as shown in the following list:

Value	Meaning
TMPF_FIXED_PITCH	Designates a fixed-pitch font.
TMPF_VECTOR	Designates a vector or TrueType font.
TMPF_TRUETYPE	Designates a TrueType font.
TMPF_DEVICE	Designates a device font.

Some fonts are identified by several of these bits--for example, the TMPF\_FIXED\_PITCH, TMPF\_VECTOR, and TMPF\_TRUETYPE bits would be set for the monospace TrueType font, Courier New®. The TMPF\_DEVICE bit could be set for a TrueType font as well, because this bit is set both for downloaded and device-resident fonts.

When the TMPF\_TRUETYPE bit is set, the font is usable on all output devices. For example, if a TrueType font existed on a printer but could not be used on the display, the TMPF\_TRUETYPE bit would not be set for that font.

The four high-order bits of this member designate the font family. The tmPitchAndFamily member can be combined with the hexadecimal value 0xF0 by using the bitwise AND operator and can then be compared with the font family names for an identical match. The following font families are defined:

Value	Meaning
FF_DECORATIVE	Novelty fonts. Old English is an example.
FF_DONTCARE	Don't care or don't know.
FF_MODERN	Fonts with constant stroke width, with or without serifs. Pica, Elite, and Courier New are examples.
FF_ROMAN	Fonts with variable stroke width and with serifs. Times New Roman and New Century Schoolbook are examples.
FF_SCRIPT	Fonts designed to look like handwriting. Script and Cursive are examples.
FF_SWISS	Fonts with variable stroke width and without serifs. <u>MS</u> Sans Serif is an example.

tmCharSet	Specifies the character set of the font. The following values are defined:												
	<table> <thead> <tr> <th>Constant</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>ANSI_CHARSET</td> <td>0</td> </tr> <tr> <td>DEFAULT_CHARSET</td> <td>1</td> </tr> <tr> <td>SYMBOL_CHARSET</td> <td>2</td> </tr> <tr> <td>SHIFTJIS_CHARSET</td> <td>128</td> </tr> <tr> <td>OEM_CHARSET</td> <td>255</td> </tr> </tbody> </table>	Constant	Value	ANSI_CHARSET	0	DEFAULT_CHARSET	1	SYMBOL_CHARSET	2	SHIFTJIS_CHARSET	128	OEM_CHARSET	255
Constant	Value												
ANSI_CHARSET	0												
DEFAULT_CHARSET	1												
SYMBOL_CHARSET	2												
SHIFTJIS_CHARSET	128												
OEM_CHARSET	255												
tmOverhang	<p>Specifies the extra width that is added to some synthesized fonts. When synthesizing some attributes, such as bold or italic, graphics device interface (GDI) or a device sometimes adds width to a string on both a per-character and per-string basis. For example, graphics device interface (GDI) makes a string bold by expanding the intracharacter spacing and overstriking by an offset value and italicizes a font by skewing the string. In either case, the string is wider after the attribute is synthesized. For bold strings, the overhang is the distance by which the overstrike is offset. For italic strings, the overhang is the amount the top of the font is skewed past the bottom of the font.</p> <p>The tmOverhang member is zero for many italic and bold TrueType fonts because many TrueType fonts include italic and bold faces that are not synthesized. For example, the overhang for Courier New®Italic is zero.</p> <p>An application that uses raster fonts can use the overhang value to determine the spacing between words that have different attributes.</p>												
tmDigitizedAspectX	Specifies the horizontal aspect of the device for which the font was designed.												
tmDigitizedAspectY	Specifies the vertical aspect of the device for which the font was designed. The ratio of the tmDigitizedAspectX and tmDigitizedAspectY members is the aspect ratio of the device for which the font was designed.												

#### Comments

All sizes are given in logical units; that is, they depend on the current mapping mode of the display context.

#### See Also

[EnumFontFamilies](#), [EnumFonts](#), [GetDeviceCaps](#), [GetTextMetrics](#), [NEWTEXTMETRIC](#)

## TIMERINFO (3.1)

```
#include <toolhelp.h>

typedef struct tagTIMERINFO { /* ti */
    DWORD dwSize;
    DWORD dwmsSinceStart;
    DWORD dwmsThisVM;
} TIMERINFO;
```

The TIMERINFO structure contains the elapsed time since the current task became active and since the virtual machine (VM) started.

Member	Description
dwSize	Specifies the size of the TIMERINFO structure, in bytes.
dwmsSinceStart	Contains the amount of time, in milliseconds, since the current task became active.
dwmsThisVM	Contains the amount of time, in milliseconds, since the current VM started.

### Comments

In standard mode, the dwmsSinceStart and dwmsThisVM values are the same.

### See Also

[TimerCount](#)



## TTPOLYCURVE (3.1)

```
typedef struct tagTTPOLYCURVE {
    UINT    wType;
    UINT    cpfx;
    POINTFX apfx[1];
} TTPOLYCURVE;
```

The TTPOLYCURVE structure contains information about a curve in the outline of a TrueType character.

Member	Description						
wType	Specifies the type of curve described by the structure. This member can be one of the following values:						
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>TT_PRIM_LINE</td><td>Curve is a polyline.</td></tr><tr><td>TT_PRIM_QSPLINE</td><td>Curve is a quadratic spline.</td></tr></tbody></table>	Value	Meaning	TT_PRIM_LINE	Curve is a polyline.	TT_PRIM_QSPLINE	Curve is a quadratic spline.
Value	Meaning						
TT_PRIM_LINE	Curve is a polyline.						
TT_PRIM_QSPLINE	Curve is a quadratic spline.						
cpfx	Specifies the number of <u>POINTFX</u> structures in the array.						
apfx	Specifies an array of <u>POINTFX</u> structures that define the polyline or quadratic spline.						

### Comments

When an application calls the GetGlyphOutline function, a glyph outline for a TrueType character is returned in a TTPOLYGONHEADER structure followed by as many TTPOLYCURVE structures as are required to describe the glyph. All points are returned as POINTFX structures and represent absolute positions, not relative moves. The starting point given by the pfxStart member of the TTPOLYGONHEADER structure is the point at which the outline for a contour begins. The TTPOLYCURVE structures that follow can be either polyline records or spline records.

Polyline records are a series of points; lines drawn between the points describe the outline of the character. Spline records represent the quadratic curves used by TrueType (that is, quadratic b-splines).

### See Also

POINTFX, TTPOLYGONHEADER, GetGlyphOutline

### TPOLYGONHEADER (3.1)

```
typedef struct tagTPOLYGONHEADER {  
    DWORD    cb;  
    DWORD    dwType;  
    POINTFX  pfxStart;  
} TPOLYGONHEADER;
```

The TPOLYGONHEADER structure specifies the starting position and type of a contour in a TrueType character outline.

Member	Description
cb	Specifies the number of bytes required by the TPOLYGONHEADER structure and <u>TPOLYCURVE</u> structure or structures required to describe the contour.
dwType	Specifies the type of character outline that is returned. Currently, this value must be TT_POLYGON_TYPE.
pfxStart	Specifies the starting point of the contour in the character outline.

#### Comments

Each TPOLYGONHEADER structure is followed by one or more TPOLYCURVE structures.

#### See Also

POINTFX, TPOLYCURVE, GetGlyphOutline

## VS\_FIXEDFILEINFO (3.1)

```
#include <ver.h>

typedef struct tagVS_FIXEDFILEINFO {    /* vsffi */
    DWORD dwSignature;
    DWORD dwStrucVersion;
    DWORD dwFileVersionMS;
    DWORD dwFileVersionLS;
    DWORD dwProductVersionMS;
    DWORD dwProductVersionLS;
    DWORD dwFileFlagsMask;
    DWORD dwFileFlags;
    DWORD dwFileOS;
    DWORD dwFileType;
    DWORD dwFileSubtype;
    DWORD dwFileDateMS;
    DWORD dwFileDateLS;
} VS_FIXEDFILEINFO;
```

The VS\_FIXEDFILEINFO structure contains version information about a file.

Member	Description
dwSignature	Specifies the value 0xFEEFO4BD.
dwStrucVersion	Specifies the binary version number of this structure. The high-order word contains the major version number, and the low-order word contains the minor version number. This value must be greater than 0x00000029.
dwFileVersionMS	Specifies the high-order 32 bits of the binary version number for the file. The value of this member is used with the value of the dwFileVersionLS member to form a 64-bit version number.
dwFileVersionLS	Specifies the low-order 32 bits of the binary version number for the file. The value of this member is used with the dwFileVersionMS value to form a 64-bit version number.
dwProductVersionMS	Specifies the high-order 32 bits of the binary version number of the product with which the file is distributed. The value of this member is used with the value of the dwProductVersionLS member to form a 64-bit version number.
dwProductVersionLS	Specifies the low-order 32 bits of the binary version number of the product with which the file is distributed. The value of this member is used with the dwProductVersionMS value to form a 64-bit version number.
dwFileFlagsMask	Specifies which bits in the dwFileFlags member are valid. If a bit is set, the corresponding bit in the dwFileFlags member is valid.
dwFileFlags	Specifies the Boolean attributes of the file. The attributes can be a combination of the following values:

Value	Meaning
VS_FF_DEBUG	File contains debugging information or is compiled with debugging features enabled.
VS_FF_INFOINFERRED	File contains a dynamically created version-information resource. Some of the blocks for the resource may be empty or incorrect. This value is not intended to be used in version-information resources created by using the <u>VERSIONINFO</u> statement.
VS_FF_PATCHED	File has been modified and is not identical to the original shipping file of the same version number.
VS_FF_PRERELEASE	File is a development version, not a commercially released product.

	VS_FF_PRIVATEBUILD	File was not built using standard release procedures. If this value is given, the StringFileInfo block must contain a PrivateBuild string.
	VS_FF_SPECIALBUILD	File was built by the original company using standard release procedures but is a variation of the standard file of the same version number. If this value is given, the StringFileInfo block must contain a SpecialBuild string.
dwFileOS	Specifies the operating system for which this file was designed. This member can be one of the following values:	
	Value	Meaning
	VOS_UNKNOWN	Operating system for which the file was designed is unknown to Windows.
	VOS_DOS	File was designed for MS-DOS.
	VOS_NT	File was designed for Windows NT.
	VOS_WINDOWS16	File was designed for Windows version 3.0 or later.
	VOS_WINDOWS32	File was designed for 32-bit Windows.
	VOS_DOS_WINDOWS16	File was designed for Windows version 3.0 or later running with MS-DOS.
	VOS_DOS_WINDOWS32	File was designed for 32-bit Windows running with MS-DOS.
	VOS_NT_WINDOWS32	File was designed for 32-bit Windows running with Windows NT.
	The values 0x00002L, 0x00003L, 0x20000L and 0x30000L are reserved.	
dwFileType	Specifies the general type of file. This type can be one of the following values:	
	Value	Meaning
	VFT_UNKNOWN	File type is unknown to Windows.
	VFT_APP	File contains an application.
	VFT_DLL	File contains a dynamic-link library (DLL).
	VFT_DRV	File contains a device driver. If the dwFileType member is VFT_DRV, the dwFileSubtype member contains a more specific description of the driver.
	VFT_FONT	File contains a font. If the dwFileType member is VFT_FONT, the dwFileSubtype member contains a more specific description of the font.
	VFT_VXD	File contains a virtual device.
	VFT_STATIC_LIB	File contains a static-link library.
	All other values are reserved for use by Microsoft.	
dwFileSubtype	Specifies the function of the file. This member is zero unless the dwFileType member is VFT_DRV, VFT_FONT, or VFT_VXD. If dwFileType is VFT_DRV, dwFileSubtype may be one of the following values:	
	Value	Meaning
	VFT2_UNKNOWN	Driver type is unknown to Windows.
	VFT2_DRV_COMM	File contains a communications driver.
	VFT2_DRV_PRINTER	File contains a printer driver.
	VFT2_DRV_KEYBOARD	File contains a keyboard driver.
	VFT2_DRV_LANGUAGE	File contains a language driver.

VFT2_DRV_DISPLAY	File contains a display driver.
VFT2_DRV_MOUSE	File contains a mouse driver.
VFT2_DRV_NETWORK	File contains a network driver.
VFT2_DRV_SYSTEM	File contains a system driver.
VFT2_DRV_INSTALLABLE	File contains an installable driver.
VFT2_DRV_SOUND	File contains a sound driver.

If dwFileType is VFT\_FONT, dwFileSubtype may be one of the following values:

Value	Meaning
VFT2_UNKNOWN	Font type is unknown to Windows.
VFT2_FONT_RASTER	File contains a raster font.
VFT2_FONT_VECTOR	File contains a vector font.
VFT2_FONT_TRUETYPE	File contains a TrueType font.

If dwFileType is VFT\_VXD, dwFileSubtype contains the virtual-device identifier included in the virtual-device control block.

All dwFileSubtype values not listed here are reserved for use by Microsoft.

dwFileDateMS	Specifies the high-order 32 bits of a binary date/time stamp for the file. The value of this member is used with the value of the dwFileDateLS member to form a 64-bit number representing the date and time the file was created.
dwFileDateLS	Specifies the low-order 32 bits of a binary date/time stamp for the file. The value of this member is used with the dwFileDateMS value to form a 64-bit number representing the date and time the file was created.

#### Comments

The binary version numbers specified in this structure are intended to be integers rather than character strings. For a file or product that has decimal points or letters in its version number, the corresponding binary version number should be a reasonable numeric representation.

A third-party developer can use the file-version values to reflect a private version-numbering scheme, as long as each new version of the product has a higher number than the previous version. The File Installation library functions use these values when comparing the ages of files.

Microsoft Windows Resource Compiler sets the dwFileDateMS and dwFileDateLS members to zero.

See Also

[VerQueryValue](#)

## WINDEBDEBUGINFO (3.1)

```
typedef struct tagWINDEBDEBUGINFO {
    UINT    flags;
    DWORD   dwOptions;
    DWORD   dwFilter;
    char    achAllocModule[8];
    DWORD   dwAllocBreak;
    DWORD   dwAllocCount;
} WINDEBDEBUGINFO;
```

The WINDEBDEBUGINFO structure contains current system-debugging information for the debugging version of Windows 3.1.

Member	Description																		
flags	Specifies which members of the WINDEBDEBUGINFO structure are valid. This member can be one or more of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>WDI_OPTIONS</td><td>dwOptions member is valid.</td></tr><tr><td>WDI_FILTER</td><td>dwFilter member is valid.</td></tr><tr><td>WDI_ALLOCBREAK</td><td>achAllocModule, dwAllocBreak, and dwAllocCount members are valid.</td></tr></tbody></table>	Value	Meaning	WDI_OPTIONS	dwOptions member is valid.	WDI_FILTER	dwFilter member is valid.	WDI_ALLOCBREAK	achAllocModule, dwAllocBreak, and dwAllocCount members are valid.										
Value	Meaning																		
WDI_OPTIONS	dwOptions member is valid.																		
WDI_FILTER	dwFilter member is valid.																		
WDI_ALLOCBREAK	achAllocModule, dwAllocBreak, and dwAllocCount members are valid.																		
dwOptions	Specifies debugging options. This member is valid only if WDI_OPTIONS is specified in the flags member. It can be one or more of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>DBO_CHECKHEAP</td><td>Performs local heap checking after all calls to functions that manipulate local memory.</td></tr><tr><td>DBO_BUFFERFILL</td><td>Fills buffers passed to API functions with 0xF9. This ensures that the supplied buffer is completely writable and helps detect overwrite problems when the supplied buffer size is not large enough.</td></tr><tr><td>DBO_DISABLEGPTTRAPPING</td><td>Disables hooking of the fault interrupt vectors. This option is not typically used by application developers, because parameter validation can cause many spurious traps that are not errors.</td></tr><tr><td>DBO_CHECKFREE</td><td>Fills all freed local memory with 0xFB. All newly allocated memory is checked to ensure that it is still filled with 0xFB--this ensures that no application has written into a freed memory object. This option has no effect if DBO_CHECKHEAP is not specified.</td></tr><tr><td>DBO_INT3BREAK</td><td>Breaks to the debugger with simple INT 3 rather than a call to the FatalExit function. This option does not generate a stack backtrace.</td></tr><tr><td>DBO_NOFATALBREAK</td><td>Does not break with the "abort, break, ignore" prompt if a DBF_FATAL message occurs.</td></tr><tr><td>DBO_NOERRORBREAK</td><td>Does not break with the "abort, break, ignore" prompt if a DBF_ERROR message occurs. This option also applies to invalid parameter errors.</td></tr><tr><td>DBO_WARNINGBREAK</td><td>Breaks with the "abort, break, ignore" prompt if a DBF_WARNING</td></tr></tbody></table>	Value	Meaning	DBO_CHECKHEAP	Performs local heap checking after all calls to functions that manipulate local memory.	DBO_BUFFERFILL	Fills buffers passed to API functions with 0xF9. This ensures that the supplied buffer is completely writable and helps detect overwrite problems when the supplied buffer size is not large enough.	DBO_DISABLEGPTTRAPPING	Disables hooking of the fault interrupt vectors. This option is not typically used by application developers, because parameter validation can cause many spurious traps that are not errors.	DBO_CHECKFREE	Fills all freed local memory with 0xFB. All newly allocated memory is checked to ensure that it is still filled with 0xFB--this ensures that no application has written into a freed memory object. This option has no effect if DBO_CHECKHEAP is not specified.	DBO_INT3BREAK	Breaks to the debugger with simple INT 3 rather than a call to the FatalExit function. This option does not generate a stack backtrace.	DBO_NOFATALBREAK	Does not break with the "abort, break, ignore" prompt if a DBF_FATAL message occurs.	DBO_NOERRORBREAK	Does not break with the "abort, break, ignore" prompt if a DBF_ERROR message occurs. This option also applies to invalid parameter errors.	DBO_WARNINGBREAK	Breaks with the "abort, break, ignore" prompt if a DBF_WARNING
Value	Meaning																		
DBO_CHECKHEAP	Performs local heap checking after all calls to functions that manipulate local memory.																		
DBO_BUFFERFILL	Fills buffers passed to API functions with 0xF9. This ensures that the supplied buffer is completely writable and helps detect overwrite problems when the supplied buffer size is not large enough.																		
DBO_DISABLEGPTTRAPPING	Disables hooking of the fault interrupt vectors. This option is not typically used by application developers, because parameter validation can cause many spurious traps that are not errors.																		
DBO_CHECKFREE	Fills all freed local memory with 0xFB. All newly allocated memory is checked to ensure that it is still filled with 0xFB--this ensures that no application has written into a freed memory object. This option has no effect if DBO_CHECKHEAP is not specified.																		
DBO_INT3BREAK	Breaks to the debugger with simple INT 3 rather than a call to the FatalExit function. This option does not generate a stack backtrace.																		
DBO_NOFATALBREAK	Does not break with the "abort, break, ignore" prompt if a DBF_FATAL message occurs.																		
DBO_NOERRORBREAK	Does not break with the "abort, break, ignore" prompt if a DBF_ERROR message occurs. This option also applies to invalid parameter errors.																		
DBO_WARNINGBREAK	Breaks with the "abort, break, ignore" prompt if a DBF_WARNING																		

		message occurs. (Normally, DBF_WARNING messages are displayed but no break occurs). This option also applies to invalid parameter warnings.
	<b><u>DBO_TRACEBREAK</u></b>	Breaks with the "abort, break, ignore" on any DBF_TRACE message that matches the value specified in the dwFilter member.
	<b><u>DBO_SILENT</u></b>	Does not display warning, error, or fatal messages except in cases where a stack trace and "abort, break, ignore" prompt would occur.
dwFilter	Specifies filtering options for DBF_TRACE messages. (Normally, trace messages are not sent to the debug terminal.) This member can be one or more of the following values:	
	Value	Meaning
	<b><u>DBF_KRN_MEMMAN</u></b>	Enables KERNEL messages related to local and global memory management.
	<b><u>DBF_KRN_LOADMODULE</u></b>	Enables KERNEL messages related to module loading.
	<b><u>DBF_KRN_SEGMENTLOAD</u></b>	Enables KERNEL messages related to segment loading.
	<b><u>DBF_APPLICATION</u></b>	Enables trace messages originating from an application.
	<b><u>DBF_DRIVER</u></b>	Enables trace messages originating from device drivers.
	<b><u>DBF_PENWIN</u></b>	Enables trace messages originating from PENWIN.
	<b><u>DBF_MMSYSTEM</u></b>	Enables trace messages originating from MMSYSTEM.
	<b><u>DBF_GDI</u></b>	Enables trace messages originating from GDI.
	<b><u>DBF_USER</u></b>	Enables trace messages originating from USER.
	<b><u>DBF_KERNEL</u></b>	Enables any trace message originating from KERNEL. (This is a combination of <b><u>DBF_KRN_MEMMAN</u></b> , <b><u>DBF_KRN_LOADMODULE</u></b> , and <b><u>DBF_KRN_SEGMENTLOAD</u></b> .)
achAllocModule	Specifies the name of the application module. (This can be different from the name of the executable file.) This cannot be the name of a dynamic-link library (DLL). The name is limited to 8 characters.	
dwAllocBreak	Specifies the number of global or local memory allocations to allow before failing allocation requests. When the count of allocations reaches the number specified in this member, that allocation and all subsequent allocations fail. If this member is zero, no allocation break is set, but the system counts allocations and reports the current count in the dwAllocCount member.	
dwAllocCount	Current count of allocations. (This information is typically retrieved by calling the <u>GetWinDebugInfo</u> function.)	

#### Comments

Developers can use the achAllocModule, dwAllocBreak, and dwAllocCount members to ensure that an application performs correctly in out-of-memory conditions. Because memory allocations made by the system fail once the break count is reached, calls to functions such as CreateWindow, CreateBrush, and SelectObject will fail as well. Only allocations made within the context of the application specified by the achAllocModule member are affected by the allocation break count.

#### See Also

DebugOutput, GetWinDebugInfo, SetWinDebugInfo

DBO\_CHECKHEAP 0x0001

Performs local heap checking after all calls to functions that manipulate local memory.



DBO\_CHECKHEAP 0x0001

DBO\_BUFFERFILL 0x0004

Fills buffers passed to API functions with 0xF9. This ensures that the supplied buffer is completely writable and helps detect overwrite problems when the supplied buffer size is not large enough.

DBO\_BUFFERFILL 0x0004

DBO\_DISABLEGPTRAPPING 0x0010

Disables hooking of the fault interrupt vectors. This option is not typically used by application developers, because parameter validation can cause many spurious traps that are not errors.

DBO\_DISABLEGPTRAPPING 0x0010

DBO\_CHECKFREE 0x0020

Fills all freed local memory with 0xFB. All newly allocated memory is checked to ensure that it is still filled with 0xFB--this ensures that no application has written into a freed memory object. This option has no effect if DBO\_CHECKHEAP is not specified.

DBO\_CHECKFREE 0x0020

DBO\_INT3BREAK 0x0100

Breaks to the debugger with simple INT 3 rather than a call to the FatalExit function. This option does not generate a stack backtrace.



DBO\_INT3BREAK 0x0100

DBO\_NOFATALBREAK 0x0400

Does not break with the "abort, break, ignore" prompt if a DBF\_FATAL message occurs.

DBO\_NOFATALBREAK 0x0400

DBO\_NOERRORBREAK 0x0800

Does not break with the "abort, break, ignore" prompt if a DBF\_ERROR message occurs. This option also applies to invalid parameter errors.

DBO\_NOERRORBREAK 0x0800

DBO\_WARNINGBREAK 0x1000

Breaks with the "abort, break, ignore" prompt if a DBF\_WARNING message occurs. (Normally, DBF\_WARNING messages are displayed but no break occurs). This option also applies to invalid parameter warnings.

DBO\_WARNINGBREAK 0x1000

DBO\_TRACEBREAK 0x2000

Breaks with the "abort, break, ignore" on any DBF\_TRACE message that matches the value specified in the dwFilter member.



DBO\_TRACEBREAK 0x2000

DBO\_SILENT 0x8000

Does not display warning, error, or fatal messages except in cases where a stack trace and "abort, break, ignore" prompt would occur.

DBO\_SILENT 0x8000

DBF\_KRN\_MEMMAN 0x0001

Enables KERNEL messages related to local and global memory management.

DBF\_KRN\_MEMMAN 0x0001

DBF\_KRN\_LOADMODULE 0x0002

Enables KERNEL messages related to module loading.

DBF\_KRN\_LOADMODULE 0x0002

DBF\_KRN\_SEGMENTLOAD 0x0004

Enables KERNEL messages related to segment loading.



DBF\_KRN\_SEGMENTLOAD 0x0004

DBF\_APPLICATION 0x0008

Enables trace messages originating from an application.

DBF\_APPLICATION 0x0008

DBF\_DRIVER 0x0010

Enables trace messages originating from device drivers.

DBF\_DRIVER 0x0010

DBF\_PENWIN 0x0020

Enables trace messages originating from PENWIN.

DBF\_PENWIN 0x0020

DBF\_MMSYSTEM 0x0040

Enables trace messages originating from MMSYSTEM.



DBF\_MMSYSTEM 0x0040

DBF\_GDI 0x0400

Enables trace messages originating from GDI.

DBF\_GDI 0x0400

DBF\_USER 0x0800

Enables trace messages originating from USER.

DBF\_USER 0x0800

DBF\_KERNEL 0x1000

Enables any trace message originating from KERNEL. (This is a combination of DBF\_KRN\_MEMMAN, DBF\_KRN\_LOADMODULE, and DBF\_KRN\_SEGMENTLOAD.)

DBF\_KERNEL 0x1000

## WINDOWPLACEMENT (3.1)

```
typedef struct tagWINDOWPLACEMENT { /* wndpl */
    UINT length;
    UINT flags;
    UINT showCmd;
    POINT ptMinPosition;
    POINT ptMaxPosition;
    RECT rcNormalPosition;
} WINDOWPLACEMENT;
```

The WINDOWPLACEMENT structure contains information about the placement of a window on the screen.

Member	Description																		
length	Specifies the length, in bytes, of the structure.																		
flags	Specifies flags that control the position of the minimized window and the method by which the window is restored. This member can be one or both of the following flags: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>WPF_SETMINPOSITION</td><td>Specifies that the x- and y-positions of the minimized window may be specified. This flag must be specified if the coordinates are set in the ptMinPosition member.</td></tr><tr><td>WPF_RESTORETOMAXIMIZED</td><td>Specifies that the restored window will be maximized, regardless of whether it was maximized before it was minimized. This setting is valid only the next time the window is restored. It does not change the default restoration behavior. This flag is valid only when the SW_SHOWMINIMIZED value is specified for the showCmd member.</td></tr></tbody></table>	Value	Meaning	WPF_SETMINPOSITION	Specifies that the x- and y-positions of the minimized window may be specified. This flag must be specified if the coordinates are set in the ptMinPosition member.	WPF_RESTORETOMAXIMIZED	Specifies that the restored window will be maximized, regardless of whether it was maximized before it was minimized. This setting is valid only the next time the window is restored. It does not change the default restoration behavior. This flag is valid only when the SW_SHOWMINIMIZED value is specified for the showCmd member.												
Value	Meaning																		
WPF_SETMINPOSITION	Specifies that the x- and y-positions of the minimized window may be specified. This flag must be specified if the coordinates are set in the ptMinPosition member.																		
WPF_RESTORETOMAXIMIZED	Specifies that the restored window will be maximized, regardless of whether it was maximized before it was minimized. This setting is valid only the next time the window is restored. It does not change the default restoration behavior. This flag is valid only when the SW_SHOWMINIMIZED value is specified for the showCmd member.																		
showCmd	Specifies the current show state of the window. This member may be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>SW_HIDE</td><td>Hides the window and passes activation to another window.</td></tr><tr><td>SW_MINIMIZE</td><td>Minimizes the specified window and activates the top-level window in the system's list.</td></tr><tr><td>SW_RESTORE</td><td>Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_SHOWNORMAL).</td></tr><tr><td>SW_SHOW</td><td>Activates a window and displays it in its current size and position.</td></tr><tr><td>SW_SHOWMAXIMIZED</td><td>Activates a window and displays it as a maximized window.</td></tr><tr><td>SW_SHOWMINIMIZED</td><td>Activates a window and displays it as an icon.</td></tr><tr><td>SW_SHOWMINNOACTIVE</td><td>Displays a window as an icon. The window that is currently active remains active.</td></tr><tr><td>SW_SHOWNA</td><td>Displays a window in its current state.</td></tr></tbody></table>	Value	Meaning	SW_HIDE	Hides the window and passes activation to another window.	SW_MINIMIZE	Minimizes the specified window and activates the top-level window in the system's list.	SW_RESTORE	Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_SHOWNORMAL).	SW_SHOW	Activates a window and displays it in its current size and position.	SW_SHOWMAXIMIZED	Activates a window and displays it as a maximized window.	SW_SHOWMINIMIZED	Activates a window and displays it as an icon.	SW_SHOWMINNOACTIVE	Displays a window as an icon. The window that is currently active remains active.	SW_SHOWNA	Displays a window in its current state.
Value	Meaning																		
SW_HIDE	Hides the window and passes activation to another window.																		
SW_MINIMIZE	Minimizes the specified window and activates the top-level window in the system's list.																		
SW_RESTORE	Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_SHOWNORMAL).																		
SW_SHOW	Activates a window and displays it in its current size and position.																		
SW_SHOWMAXIMIZED	Activates a window and displays it as a maximized window.																		
SW_SHOWMINIMIZED	Activates a window and displays it as an icon.																		
SW_SHOWMINNOACTIVE	Displays a window as an icon. The window that is currently active remains active.																		
SW_SHOWNA	Displays a window in its current state.																		



		The window that is currently active remains active.
	SW_SHOWNOACTIVATE	Displays a window in its most recent size and position. The window that is currently active remains active.
	SW_SHOWNORMAL	Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_RESTORE).
ptMinPosition	Specifies the position of the window's top-left corner when the window is minimized.	
ptMaxPosition	Specifies the position of the window's top-left corner when the window is maximized.	
rcNormalPosition	Specifies the window's coordinates when the window is in the normal (restored) position.	

See Also

[POINT](#), [RECT](#), [ShowWindow](#), [GetWindowPlacement](#), [SetWindowPlacement](#)

## WINDOWPOS (3.1)

```
typedef struct tagWINDOWPOS { /* wp */
    HWND    hwnd;
    HWND    hwndInsertAfter;
    intx;
    inty;
    intcx;
    intcy;
    UINT    flags;
} WINDOWPOS;
```

The WINDOWPOS structure contains information about the size and position of a window.

Member	Description
hwnd	Identifies the window.
hwndInsertAfter	Identifies the window behind which this window is placed.
x	Specifies the position of the left edge of the window.
y	Specifies the position of the right edge of the window.
cx	Specifies the window width.
cy	Specifies the window height.
flags	Specifies window-positioning options. This member can be one of the following values:

Value	Meaning
SWP_DRAWFRAME	Draws a frame (defined in the class description for the window) around the window. The window receives a <u>WM_NCCALCSIZE</u> message.
SWP_HIDEWINDOW	Hides the window.
SWP_NOACTIVATE	Does not activate the window.
SWP_NOMOVE	Retains current position (ignores the x and y members).
SWP_NOOWNERZORDER	Does not change the owner window's position in the Z order.
SWP_NOSIZE	Retains current size (ignores the cx and cy members).
SWP_NOREDRA	Does not redraw changes.
SWP_NOREPOSITION	Same as SWP_NOOWNERZORDER.
SWP_NOZORDER	Retains current ordering (ignores the hwndInsertAfter member).
SWP_SHOWWINDOW	Displays the window.

See Also

EndDeferWindowPos, WM\_NCCALCSIZE, WM\_WINDOWPOSCHANGED,  
WM\_WINDOWPOSCHANGING

## WNDCLASS (2.x)

```
typedef struct tagWNDCLASS {      /* wc */
    UINT style;
    WNDPROC  lpfnWndProc;
    int  cbClsExtra;
    int  cbWndExtra;
    HINSTANCE hInstance;
    HICONhIcon;
    HCURSOR  hCursor;
    HBRUSH   hbrBackground;
    LPCSTR   lpstrMenuName;
    LPCSTR   lpstrClassName;
} WNDCLASS;
```

The WNDCLASS structure contains the class attributes that are registered by the RegisterClass function.

Member	Description																						
style	Specifies the class style. These styles can be combined by using the bitwise OR operator. This can be any combination of the following values:																						
	<table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>CS_BYTEALIGNCLIENT</u></td><td>Aligns the client area of a window on the byte boundary (in the x-direction).</td></tr><tr><td><u>CS_BYTEALIGNWINDOW</u></td><td>Aligns a window on the byte boundary (in the x-direction). This flag should be set by applications that perform bitmap operations in windows by using the <u>BitBlt</u> function.</td></tr><tr><td><u>CS_CLASSDC</u></td><td>Gives the window class its own display context (shared by instances).</td></tr><tr><td><u>CS_DBLCLKS</u></td><td>Sends double-click messages to a window.</td></tr><tr><td><u>CS_GLOBALCLASS</u></td><td>Specifies that the window class is an application global class. An application global class is created by an application or library and is available to all applications. The class is destroyed when the application or library that created the class exits; it is essential, therefore, that all windows created with the application global class be closed before this occurs.</td></tr><tr><td><u>CS_HREDRAW</u></td><td>Redraws the entire window if the horizontal size changes.</td></tr><tr><td><u>CS_NOCLOSE</u></td><td>Inhibits the close option on the System menu.</td></tr><tr><td><u>CS_OWNDC</u></td><td>Gives each window instance its own display context. Note that although the <u>CS_OWNDC</u> style is convenient, it must be used with discretion because each display context occupies approximately 800 bytes of memory.</td></tr><tr><td><u>CS_PARENTDC</u></td><td>Gives the display context of the parent window to the window class.</td></tr><tr><td><u>CS_SAVEBITS</u></td><td>Specifies that the system should try to save the screen image behind a window created from this window class as a bitmap. Later, when the window is removed, the system uses the bitmap to quickly restore the screen image. This style is useful for small windows that are displayed briefly and then removed before much other screen activity takes place (for example, menus or dialog boxes). This style increases the time required to display</td></tr></tbody></table>	Value	Meaning	<u>CS_BYTEALIGNCLIENT</u>	Aligns the client area of a window on the byte boundary (in the x-direction).	<u>CS_BYTEALIGNWINDOW</u>	Aligns a window on the byte boundary (in the x-direction). This flag should be set by applications that perform bitmap operations in windows by using the <u>BitBlt</u> function.	<u>CS_CLASSDC</u>	Gives the window class its own display context (shared by instances).	<u>CS_DBLCLKS</u>	Sends double-click messages to a window.	<u>CS_GLOBALCLASS</u>	Specifies that the window class is an application global class. An application global class is created by an application or library and is available to all applications. The class is destroyed when the application or library that created the class exits; it is essential, therefore, that all windows created with the application global class be closed before this occurs.	<u>CS_HREDRAW</u>	Redraws the entire window if the horizontal size changes.	<u>CS_NOCLOSE</u>	Inhibits the close option on the System menu.	<u>CS_OWNDC</u>	Gives each window instance its own display context. Note that although the <u>CS_OWNDC</u> style is convenient, it must be used with discretion because each display context occupies approximately 800 bytes of memory.	<u>CS_PARENTDC</u>	Gives the display context of the parent window to the window class.	<u>CS_SAVEBITS</u>	Specifies that the system should try to save the screen image behind a window created from this window class as a bitmap. Later, when the window is removed, the system uses the bitmap to quickly restore the screen image. This style is useful for small windows that are displayed briefly and then removed before much other screen activity takes place (for example, menus or dialog boxes). This style increases the time required to display
Value	Meaning																						
<u>CS_BYTEALIGNCLIENT</u>	Aligns the client area of a window on the byte boundary (in the x-direction).																						
<u>CS_BYTEALIGNWINDOW</u>	Aligns a window on the byte boundary (in the x-direction). This flag should be set by applications that perform bitmap operations in windows by using the <u>BitBlt</u> function.																						
<u>CS_CLASSDC</u>	Gives the window class its own display context (shared by instances).																						
<u>CS_DBLCLKS</u>	Sends double-click messages to a window.																						
<u>CS_GLOBALCLASS</u>	Specifies that the window class is an application global class. An application global class is created by an application or library and is available to all applications. The class is destroyed when the application or library that created the class exits; it is essential, therefore, that all windows created with the application global class be closed before this occurs.																						
<u>CS_HREDRAW</u>	Redraws the entire window if the horizontal size changes.																						
<u>CS_NOCLOSE</u>	Inhibits the close option on the System menu.																						
<u>CS_OWNDC</u>	Gives each window instance its own display context. Note that although the <u>CS_OWNDC</u> style is convenient, it must be used with discretion because each display context occupies approximately 800 bytes of memory.																						
<u>CS_PARENTDC</u>	Gives the display context of the parent window to the window class.																						
<u>CS_SAVEBITS</u>	Specifies that the system should try to save the screen image behind a window created from this window class as a bitmap. Later, when the window is removed, the system uses the bitmap to quickly restore the screen image. This style is useful for small windows that are displayed briefly and then removed before much other screen activity takes place (for example, menus or dialog boxes). This style increases the time required to display																						

the window since the system must first allocate memory to store the bitmap.

### CS\_VREDRAW

Redraws the entire window if the vertical size changes.

lpfnWndProc	Points to the window procedure. For more information, see the description of the <u>WindowProc</u> callback function.																				
cbClsExtra	Specifies the number of bytes to allocate following the window-class structure. These bytes are initialized to zero.																				
cbWndExtra	Specifies the number of bytes to allocate following the window instance. These bytes are initialized to zero. If an application uses the WNDCLASS structure to register a dialog box created with the CLASS directive in the resource file, it must set this member to DLGWINDOWEXTRA.																				
hInstance	Identifies the class module. This member must be an instance handle and must not be NULL.																				
hIcon	Identifies the class icon. This member must be a handle to an icon resource. If this member is NULL, the application must draw an icon whenever the user minimizes the application's window.																				
hCursor	Identifies the class cursor. This member must be a handle to a cursor resource. If this member is NULL, the application must explicitly set the cursor shape whenever the mouse moves into the application's window.																				
hbrBackground	Identifies the class background brush. This member can be either a handle to the physical brush that is to be used for painting the background, or it can be a color value. If a color value is given, it must be one of the standard system colors listed below, and the value 1 must be added to the chosen color (for example, <u>COLOR_BACKGROUND</u> + 1 specifies the system background color). If a color value is given, it must be converted to one of the following HBRUSH types: <table><tr><td><u>COLOR_ACTIVEBORDER</u></td><td><u>COLOR_HIGHLIGHTTEXT</u></td></tr><tr><td><u>COLOR_ACTIVECAPTION</u></td><td><u>COLOR_INACTIVEBORDER</u></td></tr><tr><td><u>COLOR_APPWORKSPACE</u></td><td><u>COLOR_INACTIVECAPTION</u></td></tr><tr><td><u>COLOR_BACKGROUND</u></td><td><u>COLOR_INACTIVECAPTIONTEXT</u></td></tr><tr><td><u>COLOR_BTNFACE</u></td><td><u>COLOR_MENU</u></td></tr><tr><td><u>COLOR_BTNSHADOW</u></td><td><u>COLOR_MENUTEXT</u></td></tr><tr><td><u>COLOR_BTNTEXT</u></td><td><u>COLOR_SCROLLBAR</u></td></tr><tr><td><u>COLOR_CAPTIONTEXT</u></td><td><u>COLOR_WINDOW</u></td></tr><tr><td><u>COLOR_GRAYTEXT</u></td><td><u>COLOR_WINDOWFRAME</u></td></tr><tr><td><u>COLOR_HIGHLIGHT</u></td><td><u>COLOR_WINDOWTEXT</u></td></tr></table> <p>The system automatically deletes class background brushes when the class is freed. An application should not delete these brushes because a class may be used by multiple instances of the application.</p> <p>When this member is NULL, the application must paint its own background whenever it is requested to paint in its client area. The application can determine when the background needs painting by processing the <u>WM_ERASEBKGD</u> message or by testing the fErase member of the <u>PAINTSTRUCT</u> structure filled by the <u>BeginPaint</u> function.</p>	<u>COLOR_ACTIVEBORDER</u>	<u>COLOR_HIGHLIGHTTEXT</u>	<u>COLOR_ACTIVECAPTION</u>	<u>COLOR_INACTIVEBORDER</u>	<u>COLOR_APPWORKSPACE</u>	<u>COLOR_INACTIVECAPTION</u>	<u>COLOR_BACKGROUND</u>	<u>COLOR_INACTIVECAPTIONTEXT</u>	<u>COLOR_BTNFACE</u>	<u>COLOR_MENU</u>	<u>COLOR_BTNSHADOW</u>	<u>COLOR_MENUTEXT</u>	<u>COLOR_BTNTEXT</u>	<u>COLOR_SCROLLBAR</u>	<u>COLOR_CAPTIONTEXT</u>	<u>COLOR_WINDOW</u>	<u>COLOR_GRAYTEXT</u>	<u>COLOR_WINDOWFRAME</u>	<u>COLOR_HIGHLIGHT</u>	<u>COLOR_WINDOWTEXT</u>
<u>COLOR_ACTIVEBORDER</u>	<u>COLOR_HIGHLIGHTTEXT</u>																				
<u>COLOR_ACTIVECAPTION</u>	<u>COLOR_INACTIVEBORDER</u>																				
<u>COLOR_APPWORKSPACE</u>	<u>COLOR_INACTIVECAPTION</u>																				
<u>COLOR_BACKGROUND</u>	<u>COLOR_INACTIVECAPTIONTEXT</u>																				
<u>COLOR_BTNFACE</u>	<u>COLOR_MENU</u>																				
<u>COLOR_BTNSHADOW</u>	<u>COLOR_MENUTEXT</u>																				
<u>COLOR_BTNTEXT</u>	<u>COLOR_SCROLLBAR</u>																				
<u>COLOR_CAPTIONTEXT</u>	<u>COLOR_WINDOW</u>																				
<u>COLOR_GRAYTEXT</u>	<u>COLOR_WINDOWFRAME</u>																				
<u>COLOR_HIGHLIGHT</u>	<u>COLOR_WINDOWTEXT</u>																				
lpzMenuName	Points to a null-terminated string that specifies the resource name of the class menu (as the name appears in the resource file). If an integer is used to identify the menu, the <u>MAKEINTRESOURCE</u> macro can be used. If this member is NULL, windows belonging to this class have no default menu.																				
lpzClassName	Points to a null-terminated string that specifies the name of the window class.																				

See Also

PAINTSTRUCT, MAKEINTRESOURCE, RegisterClass, WindowProc

CS\_BYTEALIGNCLIENT 0x1000

Aligns the client area of a window on the byte boundary (in the x-direction).

CS\_BYTEALIGNCLIENT 0x1000

CS\_BYTEALIGNWINDOW 0x2000

Aligns a window on the byte boundary (in the x-direction). This flag should be set by applications that perform bitmap operations in windows by using the BitBlt function.

CS\_BYTEALIGNWINDOW 0x2000



CS\_CLASSDC 0x0040

Gives the window class its own display context (shared by instances).

CS\_CLASSDC 0x0040

CS\_DBLCLKS 0x0008

Sends double-click messages to a window.

CS\_DBLCLKS 0x0008

CS\_GLOBALCLASS 0x4000

Specifies that the window class is an application global class. An application global class is created by an application or library and is available to all applications. The class is destroyed when the application or library that created the class exits; it is essential, therefore, that all windows created with the application global class be closed before this occurs.

CS\_GLOBALCLASS 0x4000

CS\_HREDRAW 0x0002

Redraws the entire window if the horizontal size changes.

CS\_HREDRAW 0x0002



CS\_NOCLOSE 0x0200

Inhibits the close option on the System menu.

CS\_NOCLOSE 0x0200

CS\_OWNDC 0x0020

Gives each window instance its own display context. Note that although the CS\_OWNDC style is convenient, it must be used with discretion because each display context occupies approximately 800 bytes of memory.

CS\_OWDC 0x0020

CS\_PARENTDC 0x0080

Gives the display context of the parent window to the window class.

CS\_PARENTDC 0x0080

`CS_SAVEBITS 0x0800`

Specifies that the system should try to save the screen image behind a window created from this window class as a bitmap. Later, when the window is removed, the system uses the bitmap to quickly restore the screen image. This style is useful for small windows that are displayed briefly and then removed before much other screen activity takes place (for example, menus or dialog boxes). This style increases the time required to display the window since the system must first allocate memory to store the bitmap.

CS\_SAVEBITS 0x0800



CS\_VREDRAW 0x0001

Redraws the entire window if the vertical size changes.

CS\_VREDRAW 0x0001

### Windows structures (3.1)

<u>ABC</u>	Contains width of a character in a TrueType font
<u>BITMAP</u>	Defines characteristics of a logical bitmap
<u>BITMAPCOREHEADER</u>	Defines characteristics of a DIB
<u>BITMAPCOREINFO</u>	Defines characteristics and colors of a DIB
<u>BITMAPFILEHEADER</u>	Defines characteristics of a DIB file
<u>BITMAPINFO</u>	Defines characteristics and colors of a DIB
<u>BITMAPINFOHEADER</u>	Defines characteristics of a DIB
<u>CBT_CREATEWND</u>	Contains data passed to a <u>WH_CBT</u> hook function
<u>CBTACTIVATESTRUCT</u>	Contains data passed to a <u>WH_CBT</u> hook function
<u>CHOOSECOLOR</u>	Contains data for the color-selection dialog box
<u>CHOOSEFONT</u>	Contains data for the font-selection dialog box
<u>CLASSENTRY</u>	Contains the name of a Windows class
<u>CLIENTCREATESTRUCT</u>	Defines Window menu and first MDI child window
<u>COMPAREITEMSTRUCT</u>	Contains data for a sorted owner-drawn combo box
<u>COMSTAT</u>	Contains data about a communications device
<u>CONVCONTEXT</u>	Contains language data for a DDE conversation
<u>CONVINFO</u>	Contains information about a DDE conversation
<u>CPLINFO</u>	Contains resource data for Control Panel application
<u>CREATESTRUCT</u>	Defines window-initialization parameters
<u>CTLINFO</u>	Defines class name and version of selected control
<u>CTLSTYLE</u>	Specifies attributes of selected control
<u>CTLTYPE</u>	Specifies width, height, and style of control
<u>DCB</u>	Defines settings for serial communications device
<u>DDEACK</u>	Contains status flags sent by a <u>WM_DDE_ACK</u> message
<u>DDEADVISE</u>	Contains flags sent by a <u>WM_DDE_ADVISE</u> message
<u>DDEDATA</u>	Contains data sent by a <u>WM_DDE_DATA</u> message
<u>DDEPOKE</u>	Contains data sent by a <u>WM_DDE_POKE</u> message
<u>DEBUGHOOKINFO</u>	Contains data used for debugging
<u>DELETEITEMSTRUCT</u>	Describes a deleted owner-drawn item
<u>DEVMODE</u>	Contains information about the printer environment
<u>DEVNAMES</u>	Contains device data for the Print dialog box
<u>DOCINFO</u>	Contains document input and output filenames
<u>DRAWITEMSTRUCT</u>	Contains painting data for an owner-drawn control
<u>DRIVERINFOSTRUCT</u>	Contains data about an installable driver
<u>DRVCONFIGINFO</u>	Contains data about the configuration of a driver
<u>EVENTMSG</u>	Contains message information for a journaling hook
<u>FINDREPLACE</u>	Contains data for a Find or Replace dialog box
<u>FIXED</u>	Contains integral and fractional parts of a number
<u>FMS_GETDRIVEINFO</u>	Contains drive data for File Manager
<u>FMS_GETFILESEL</u>	Contains file data for File Manager
<u>FMS_LOAD</u>	Contains custom menu data for File Manager
<u>GLOBALENTRY</u>	Describes a memory object on the global heap
<u>GLOBALINFO</u>	Describes the global heap
<u>GLYPHMETRICS</u>	Describes placement of a glyph in a character cell
<u>HANDLETABLE</u>	Contains an array of handles to GDI objects
<u>HARDWAREHOOKSTRUCT</u>	Contains data about a nonstandard hardware message
<u>HELPWININFO</u>	Contains message information for a journaling hook
<u>HSZPAIR</u>	Contains a DDE service name and topic name
<u>KERNINGPAIR</u>	Defines a kerning pair
<u>LOCALENTRY</u>	Describes a memory object on the local heap
<u>LOCALINFO</u>	Describes the local heap
<u>LOGBRUSH</u>	Defines characteristics of a logical brush
<u>LOGFONT</u>	Specifies attributes of a logical font
<u>LOGPALETTE</u>	Defines a logical color palette
<u>LOGPEN</u>	Defines characteristics of a logical pen
<u>MAT2</u>	Contains values for a transformation matrix
<u>MDICREATESTRUCT</u>	Contains initialization data for MDI child window
<u>MEASUREITEMSTRUCT</u>	Contains dimensions of an owner-drawn control
<u>MEMMANINFO</u>	Describes the status of the virtual-memory manager
<u>MENUITEMTEMPLATE</u>	Defines a menu item
<u>MENUITEMTEMPLATEHEADER</u>	Contains header data for a menu template
<u>METAFILEPICT</u>	Defines metafile picture format for clipboard
<u>METAHEADER</u>	Contains information about a metafile

<u>METARECORD</u>	Contains a metafile record
<u>MINMAXINFO</u>	Contains window size and tracking data
<u>MODULEENTRY</u>	Describes a module in the module list
<u>MONCBSTRUCT</u>	Contains data about the current DDE transaction
<u>MONCONVSTRUCT</u>	Contains data about a DDE conversation
<u>MONERRSTRUCT</u>	Contains data about the current DDE error
<u>MONHSZSTRUCT</u>	Contains data about a DDE string handle
<u>MONLINKSTRUCT</u>	Contains data about a DDE advise loop
<u>MONMSGSTRUCT</u>	Contains data about a DDE message
<u>MOUSEHOOKSTRUCT</u>	Contains data about a mouse event
<u>MSG</u>	Contains message information
<u>MULTIKEYHELP</u>	Contains keyword data for Windows Help
<u>NCCALCSIZE_PARAMS</u>	Contains data for calculating client area
<u>NEWCPINFO</u>	Contains resource data of Control Panel application
<u>NEWTEXTMETRIC</u>	Contains basic information about a physical font
<u>NEYLOADSEG</u>	Describes the segment being loaded
<u>NEYLOGERROR</u>	Describes a validation error
<u>NEYLOGPARAMERROR</u>	Describes a parameter-validation error
<u>NEYRIP</u>	Contains the RIP exit code and relevant registers
<u>NEYSTARTDLL</u>	Describes the dynamic-link library being loaded
<u>OESTRUCT</u>	Contains data about an open file
<u>OLECLIENT</u>	Points to structure providing state information
<u>OLECLIENTVTBL</u>	Points to client's callback function
<u>OLEOBJECT</u>	Points to table of object-function pointers
<u>OLEOBJECTVTBL</u>	Points to functions for object manipulation
<u>OLESERVER</u>	Points to table of server-function pointers
<u>OLESERVERDOC</u>	Points to table of document-function pointers
<u>OLESERVERDOCVTBL</u>	Points to functions for document manipulation
<u>OLESERVERVTBL</u>	Points to functions for server manipulation
<u>OLESTREAM</u>	Points to structure providing stream functions
<u>OLESTREAMVTBL</u>	Points to functions for stream operations
<u>OLETARGETDEVICE</u>	Contains information about target device for client
<u>OPENFILENAME</u>	Contains data for the Open dialog box
<u>OUTLINETEXTMETRIC</u>	Contains TrueType font metrics
<u>PAINTSTRUCT</u>	Contains painting data for a client area
<u>PALETTEENTRY</u>	Specifies an entry in a logical color palette
<u>PANOSE</u>	Contains Panose values for a TrueType font
<u>POINT</u>	Contains the coordinates of a point
<u>POINTEX</u>	Describes a point in a character outline
<u>PRINTDLG</u>	Contains data for the Print dialog box
<u>RASTERIZER_STATUS</u>	Contains data about TrueType installation
<u>RECT</u>	Defines the coordinates of a rectangle
<u>RGBQUAD</u>	Describes colors for a DIB
<u>RGBTRIPLE</u>	Describes colors for a DIB
<u>SEGINFO</u>	Contains code- or data-segment information
<u>SIZE</u>	Contains extents when a function returns
<u>STACKTRACEENTRY</u>	Describes one stack frame
<u>SYSHEAPINFO</u>	Describes the User and GDI modules
<u>TASKENTRY</u>	Contains information about a task
<u>TEXTMETRIC</u>	Contains information about a physical font
<u>TIMERINFO</u>	Contains elapsed execution times of a task and VM
<u>TPOLYCURVE</u>	Describes a curve in a character outline
<u>TPOLYGONHEADER</u>	Specifies starting point for character outline
<u>VS_FIXEDFILEINFO</u>	Contains version information about a file
<u>WINDEBUGINFO</u>	Contains system-debugging information
<u>WINDOWPLACEMENT</u>	Contains window-placement information
<u>WINDOWPOS</u>	Contains window size and position information
<u>WNDCLASS</u>	Defines attributes of a window class

## Database Tables (3.1)

Binary and Ternary Raster-Operation Codes

Clipboard formats

Control classes

Control styles

Error Values

Metafile Records

Module and Library Names

Naming Conventions

Resource Compiler Diagnostic Messages

Virtual Key Codes

## Control styles (3.1)

Button styles

Combination box styles

Edit control styles

List box styles

Scroll bar styles

Static control styles

## MS Windows Naming Conventions

The following examples show some of the standard prefix and base types you will see in this database:

ABORTPROC	abrtprc
ATOM	atm
BOOL	f
BYTE	b
BYTE FAR*	lpb
char FAR*	lpch
DLGPROC	dlgprc
DWORD	dw
DWORD FAR*	lpdw
EDITWORDBREAKPROC	ewbprc
ENUMPROPPROC	enmprc
FONTENUMPROC	fntenmprc
GNOTIFYPROC	gnprc
GOBJENUMPROC	goenmprc
GRAYSTRINGPROC	gsprc
HACCEL	haccl
HBITMAP	hbm
HBRUSH	hbr
HCURSOR	hcur
HDC	hdc
HDRVR	hdrvr
HDWP	hdwp
HFILE	hf
HFONT	hfont
HGDIOBJ	hgdiobj
HGLOBAL	hglb
HHOOK	hhook
HICON	hicon
HINSTANCE	hinst
HLOCAL	hloc
HMENU	hmenu
HMETAFILE	hmf
HMODULE	hmod
HOOKPROC	hkprc
HPALETTE	hpal
HPEN	hpen
HRGN	hrgn
HRSRC	hrsrc
HSTR	hstr
HTASK	htask
HWND	hwnd
int	n (optional)
LINEDDAPROC	lnddaprc
LNOTIFYPROC	lnmprc
LONG	l
LPARAM	lParam
LPBYTE	lpb
LPCSTR	lpsz

LPINT	lpn
LPLONG	lpl
LPSTR	lpstr
LPVOID	lpv
LPWORD	lpw
LRESULT	lResult
MFENUMPROC	mfenmproc
NPSTR	npsz
PBYTE	npb
POINT FAR*	lppt
PROPENUMPROC	prpenmproc
RECT FAR*	lprc
RSRCHDLRPROC	rschldprc
TIMERPROC	tmprc
UINT	u (optional)
WNDENUMPROC	wndenmproc
WNDPROC	wndprc
WORD	u or w
WPARAM	wParam



### Clipboard formats (3.1)

Value	Meaning
CF_BITMAP	The data is a bitmap.
CF_DIB	The data is a memory object containing a <u>BITMAPINFO</u> structure followed by the bitmap data.
CF_DIF	The data is in Data Interchange Format (DIF).
CF_DSPBITMAP	The data is a bitmap representation of a private format. This data is displayed in bitmap format in lieu of the privately formatted data.
CF_DSPMETAFILEPICT	The data is a metafile representation of a private data format. This data is displayed in metafile-picture format in lieu of the privately formatted data.
CF_DSPTTEXT	The data is a textual representation of a private data format. This data is displayed in text format in lieu of the privately formatted data.
CF_METAFILEPICT	The data is a metafile (see the description of the <u>METAFILEPICT</u> structure).
CF_OEMTEXT	The data is an array of text characters in the OEM character set. Each line ends with a carriage return– linefeed (CR-LF) combination. A null character signals the end of the data.
CF_OWNERDISPLAY	The data is in a private format that the clipboard owner must display.
CF_PALETTE	The data is a color palette.
CF_PENDATA	The data is for the pen extensions to the Windows operating system.
CF_RIFF	The data is in Resource Interchange File Format (RIFF).
CF_SYLK	The data is in Microsoft Symbolic Link (SYLK) format.
CF_TEXT	The data is an array of text characters. Each line ends with a carriage return– linefeed (CR-LF) combination. A null character signals the end of the data.
CF_TIFF	The data is in Tag Image File Format (TIFF).
CF_WAVE	The data describes a sound wave. This is a subset of the CF_RIFF data format; it can be used only for RIFF WAVE files.

### Control classes (3.1)

Class	Description
BUTTON	A button control is a small rectangular child window that represents a "button" the user can turn on or off by clicking it with the mouse. Button controls can be used alone or in groups and can either be labeled or appear without text. Button controls typically change appearance when the user clicks them.
COMBOBOX	<p>A combo box control consists of a text box similar to an edit control, plus a list box. The list box may be displayed at all times or may be dropped down when the user selects a "pop box" next to the text box.</p> <p>The style of the combo box determines whether the user can edit the contents of the text box. If the list box is visible, typing characters into the text box causes the first list box entry that matches the characters typed to be highlighted. Conversely, selecting an item in the list box displays the selected text in the text box.</p>
EDIT	<p>An edit control is a rectangular child window in which the user can enter text from the keyboard. The user selects the control and gives it the input focus by clicking the mouse inside it or pressing the TAB key. The user can enter text when the control displays a flashing caret. The mouse can be used to move the cursor and select characters to be replaced or to position the cursor for inserting characters. The BACKSPACE key can be used to delete characters.</p> <p>Edit controls expand tab characters into as many space characters as are required to move the cursor to the next tab stop. The default for tab stops is eight characters.</p>
LISTBOX	A list box control consists of a list of items. The control is used whenever an application needs to present a list of names, such as filenames, that the user can view and select. The user can select an item by pointing to the name with the mouse and clicking a mouse button. When an item is selected, it is highlighted, and a notification message is passed to the parent window. A scroll bar can be used with a list box control to scroll lists that are too long or too wide for the control window.
SCROLLBAR	<p>A scroll bar control is a rectangle that contains a scroll box and has direction arrows at both ends. The scroll bar sends a notification message to its parent whenever the user clicks the mouse in the control. The parent is responsible for updating the scroll box position, if necessary. Scroll bar controls have the same appearance and function as the scroll bars used in ordinary windows. But unlike scroll bars, scroll bar controls can be positioned anywhere within a window and used whenever needed to provide scrolling input for a window.</p> <p>The scroll bar class also includes size box controls. A size box control is a small rectangle that the user can expand to change the size of the window.</p>
STATIC	A static control is a simple text field, box, or rectangle that can be used to label, box, or separate other controls. Static controls take no input and provide no output.

## Virtual Key Codes

The following table shows the symbolic constant names, hexadecimal values, and keyboard equivalents for the virtual-key codes used by the Microsoft Windows operating system version 3.1. The codes are listed in numeric order.

Symbolic name	Value (in hex)	Mouse or keyboard equivalent
VK_LBUTTON	01	Left mouse button
VK_RBUTTON	02	Right mouse button
VK_CANCEL	03	Used for control-break processing
VK_MBUTTON	04	Middle mouse button (three-button mouse)
--	05-07	Undefined
VK_BACK	08	BACKSPACE key
VK_TAB	09	TAB key
--	0A- 0B	Undefined
VK_CLEAR	0C	CLEAR key
VK_RETURN	0D	ENTER key
--	0E- 0F	Undefined
VK_SHIFT	10	SHIFT key
VK_CONTROL	11	CTRL key
VK_MENU	12	ALT key
VK_PAUSE	13	PAUSE key
VK_CAPITAL	14	CAPS LOCK key
--	15-19	Reserved for Kanji systems
--	1A	Undefined
VK_ESCAPE	1B	ESC key
--	1C- 1F	Reserved for Kanji systems
VK_SPACE	20	SPACEBAR
VK_PRIOR	21	PAGE UP key
VK_NEXT	22	PAGE DOWN key
VK_END	23	END key
VK_HOME	24	HOME key
VK_LEFT	25	LEFT ARROW key
VK_UP	26	UP ARROW key
VK_RIGHT	27	RIGHT ARROW key
VK_DOWN	28	DOWN ARROW key
VK_SELECT	29	SELECT key
--	2A	OEM specific
VK_EXECUTE	2B	EXECUTE key
VK_SNAPSHOT	2C	PRINT SCREEN key for Windows 3.0 and later
VK_INSERT	2D	INS key
VK_DELETE	2E	DEL key
VK_HELP	2F	HELP key
VK_0	30	0 key
VK_1	31	1 key
VK_2	32	2 key
VK_3	33	3 key
VK_4	34	4 key
VK_5	35	5 key
VK_6	36	6 key
VK_7	37	7 key
VK_8	38	8 key
VK_9	39	9 key
--	3A- 40	Undefined

VK_A	41	A key
VK_B	42	B key
VK_C	43	C key
VK_D	44	D key
VK_E	45	E key
VK_F	46	F key
VK_G	47	G key
VK_H	48	H key
VK_I	49	I key
VK_J	4A	J key
VK_K	4B	K key
VK_L	4C	L key
VK_M	4D	M key
VK_N	4E	N key
VK_O	4F	O key
VK_P	50	P key
VK_Q	51	Q key
VK_R	52	R key
VK_S	53	S key
VK_T	54	T key
VK_U	55	U key
VK_V	56	V key
VK_W	57	w key
VK_X	58	X key
VK_Y	59	Y key
VK_Z	5A	Z key
--	5B– 5F	Undefined
VK_NUMPAD0	60	Numeric keypad 0 key
VK_NUMPAD1	61	Numeric keypad 1 key
VK_NUMPAD2	62	Numeric keypad 2 key
VK_NUMPAD3	63	Numeric keypad 3 key
VK_NUMPAD4	64	Numeric keypad 4 key
VK_NUMPAD5	65	Numeric keypad 5 key
VK_NUMPAD6	66	Numeric keypad 6 key
VK_NUMPAD7	67	Numeric keypad 7 key
VK_NUMPAD8	68	Numeric keypad 8 key
VK_NUMPAD9	69	Numeric keypad 9 key
VK_MULTIPLY	6A	Multiply key
VK_ADD	6B	Add key
VK_SEPARATOR	6C	Separator key
VK_SUBTRACT	6D	Subtract key
VK_DECIMAL	6E	Decimal key
VK_DIVIDE	6F	Divide key
VK_F1	70	F1 key
VK_F2	71	F2 key
VK_F3	72	F3 key
VK_F4	73	F4 key
VK_F5	74	F5 key
VK_F6	75	F6 key
VK_F7	76	F7 key
VK_F8	77	F8 key
VK_F9	78	F9 key

VK_F10	79	F10 key
VK_F11	7A	F11 key
VK_F12	7B	F12 key
VK_F13	7C	F13 key
VK_F14	7D	F14 key
VK_F15	7E	F15 key
VK_F16	7F	F16 key
VK_F17	80H	F17 key
VK_F18	81H	F18 key
VK_F19	82H	F19 key
VK_F20	83H	F20 key
VK_F21	84H	F21 key
VK_F22	85H	F22 key
VK_F23	86H	F23 key
VK_F24	87H	F24 key
--	88-8F	Unassigned
VK_NUMLOCK	90	NUM LOCK key
VK_SCROLL	91	SCROLL LOCK key
--	92- B9	Unassigned
--	BA- C0	OEM specific
--	C1- DA	Unassigned
--	DB- E4	OEM specific
--	E5	Unassigned
--	E6	OEM specific
--	E7- E8	Unassigned
--	E9- F5	OEM specific
--	F6- FE	Unassigned

ATOM

16-bit value used as an atom handle.

BOOL

16-bit Boolean value.

## BYTE

8-bit unsigned integer. Use LPBYTE to create 32-bit pointers. Use PBYTE to create pointers that match the compiler memory model.



CATCHBUF[9]

18-byte buffer used by the Catch function.

**COLORREF**

32-bit value used as a color value.

DLGPROC

32-bit pointer to a dialog box procedure.

## DWORD

32-bit unsigned integer or a segment:offset address. Use LPDWORD to create 32-bit pointers. Use PDWORD to create pointers that match the compiler memory model.

FARPROC

32-bit pointer to a function.

**FNCALLBACK**

32-bit value identifying the DdeCallback function. Use PFNCALLBACK to create pointers that match the compiler memory model.

FONTENUMPROC

32-bit pointer to an EnumFontsProc callback function.

GNOTIFYPROC

32-bit pointer to a NotifyProc callback function.



GOBJENUMPROC

32-bit pointer to a EnumObjectsProc callback function.

## HANDLE

16-bit value used as a general handle. Use LPHANDLE to create 32-bit pointers. Use SPHANDLE to create 16-bit pointers. Use PHANDLE to create pointers that match the compiler memory model.

**HCURSOR**

16-bit value used as a cursor handle.

**HFILE**

16-bit value used as a file handle.

**HGDIOBJ**

16-bit value used as a graphics device interface (GDI) object handle.

**HGLOBAL**

16-bit value used as a handle to a global memory object.

## HHOOK

32-bit value used as a hook handle.

## HKEY

32-bit value used as a handle to a key in the registration database. Use PHKEY to create 32-bit pointers.



## HINSTANCE

16-bit handle to an instance of a module or application.

## HLOCAL

16-bit value used as a handle to a local memory object.

**HMODULE**

16-bit value used as a module handle.

## HOBJECT

16-bit value used as a handle to an OLE object.

**HWND**

16-bit value used as a handle to a window.

**HOOKPROC**

32-bit pointer to a hook procedure.

**HRSRC**

16-bit value used as a resource handle.

LHCLIENTDOC

32-bit value used as a handle to an OLE client document.



LHSERVER

32-bit value used as a handle to an OLE server.

LHSERVERDOC

32-bit value used as a handle to an OLE server document.

LONG

32-bit signed integer.

LPABC

32-bit pointer to an ABC structure.

## LPARAM

32-bit signed value passed as a parameter to a window procedure or callback function.

LPBI

32-bit pointer to a BANDINFOSTRUCT structure.

## LPBITMAP

32-bit pointer to a BITMAP structure. Use NPBITMAP to create 16-bit pointers. Use PBITMAP to create pointers that match the compiler memory model.

LPBITMAPCOREHEADER

32-bit pointer to a BITMAPCOREHEADER structure. Use PBITMAPCOREHEADER to create pointers that match the compiler memory model.



LPBITMAPCOREINFO

32-bit pointer to a BITMAPCOREINFO structure. Use PBITMAPCOREINFO to create pointers that match the compiler memory model.

## LPBITMAPFILEHEADER

32-bit pointer to a BITMAPFILEHEADER structure. Use PBITMAPFILEHEADER to create pointers that match the compiler memory model.

LPBITMAPINFO

32-bit pointer to a BITMAPINFO structure. Use PBITMAPINFO to create pointers that match the compiler memory model.

## LPBITMAPINFOHEADER

32-bit pointer to a BITMAPINFOHEADER structure. Use PBITMAPINFOHEADER to create pointers that match the compiler memory model.

LPCATCHBUF

32-bit pointer to a CATCHBUF array.

LPCBT\_CREATEWND

32-bit pointer to a CBT\_CREATEWND structure.

LPCHOOSECOLOR

32-bit pointer to a CHOOSECOLOR structure.

LPCHOOSEFONT

32-bit pointer to a CHOOSEFONT structure.



LPCLIENTCREATESTRUCT

32-bit pointer to a CLIENTCREATESTRUCT structure.

## LPCOMPAREITEMSTRUCT

32-bit pointer to a COMPAREITEMSTRUCT structure. Use PCOMPAREITEMSTRUCT to create pointers that match the compiler memory model.

LPCPLINFO

32-bit pointer to a CPLINFO structure. Use PCPLINFO to create pointers that match the compiler memory model.

LPCREATESTRUCT

32-bit pointer to a CREATESTRUCT structure.

LPCSTR

32-bit pointer to a nonmodifiable character string.

LPCTLINFO

32-bit pointer to a CTLINFO structure. Use PCTLINFO to create pointers that match the compiler memory model.

LPCTLSTYLE

32-bit pointer to a CTLSTYLE structure. Use PCTLSTYLE to create pointers that match the compiler memory model.

LPDCB

32-bit pointer to a DCB structure.



LPDEBUGHOOKINFO

32-bit pointer to a DEBUGHOOKINFO structure.

## LPDELETEITEMSTRUCT

32-bit pointer to a DELETEITEMSTRUCT structure. Use PDELETEITEMSTRUCT to create pointers that match the compiler memory model.

## LPDEVMODE

32-bit pointer to a DEVMODE structure. Use NPDEVMODE to create 16-bit pointers. Use PDEVMODE to create pointers that match the compiler memory model.

LPDEVNAMES

32-bit pointer to a DEVNAMES structure.

LPDOCINFO

32-bit pointer to a DOCINFO structure.

## LPDRAWITEMSTRUCT

32-bit pointer to a DRAWITEMSTRUCT structure. Use PDRAWITEMSTRUCT to create pointers that match the compiler memory model.

LPDRIVERINFOSTRUCT

32-bit pointer to a DRIVERINFOSTRUCT structure.

## LPDRVCONFIGINFO

32-bit pointer to a DRVCONFIGINFO structure. Use PDRVCONFIGINFO to create pointers that match the compiler memory model.



## LPEVENTMSG

32-bit pointer to a EVENTMSG structure. Use NPEVENTMSG to create 16-bit pointers. Use PEVENTMSG to create pointers that match the compiler memory model.

LPFINDREPLACE

32-bit pointer to a FINDREPLACE structure.

LPFMS\_GETDRIVEINFO

32-bit pointer to a FMS\_GETDRIVEINFO structure.

LPFMS\_GETFILESEL

32-bit pointer to a FMS\_GETFILESEL structure.

LPFMS\_LOAD

32-bit pointer to a FMS\_LOAD structure.

## LPHANDLETABLE

32-bit pointer to a HANDLETABLE structure. Use PHANDLETABLE to create pointers that match the compiler memory model.

## LPHELPWININFO

32-bit pointer to a HELPWININFO structure. Use PHELPWININFO to create pointers that match the compiler memory model.

LPINT

32-bit pointer to a 16-bit signed value. Use PINT to create pointers that match the compiler memory model.



LPKERNINGPAIR

32-bit pointer to a KERNINGPAIR structure.

## LPLOGBRUSH

32-bit pointer to a LOGBRUSH structure. Use NPLOGBRUSH to create 16-bit pointers. Use PLOGBRUSH to create pointers that match the compiler memory model.

## LPLOGFONT

32-bit pointer to a LOGFONT structure. Use NPLOGFONT to create 16-bit pointers. Use PLOGFONT to create pointers that match the compiler memory model.

LPLOGPALETTE

32-bit pointer to a LOGPALETTE structure. Use NPLOGPALETTE to create 16-bit pointers. Use PLOGPALETTE to create pointers that match the compiler memory model.

## LPLOGPEN

32-bit pointer to a LOGPEN structure. Use NPLOGPEN to create 16-bit pointers. Use PLOGPEN to create pointers that match the compiler memory model.

LPLONG

32-bit pointer to a 32-bit signed integer. Use PLONG to create pointers that match the compiler memory model.

LPMAT2

32-bit pointer to a MAT2 structure.

LPMDICREATESTRUCT

32-bit pointer to an MDICREATESTRUCT structure.



## LPMEASUREITEMSTRUCT

32-bit pointer to a MEASUREITEMSTRUCT structure. Use PMEASUREITEMSTRUCT to create pointers that match the compiler memory model.

LPMETAFILEPICK

32-bit pointer to a METAFILEPICK structure.

## LPMETARECORD

32-bit pointer to a METARECORD structure. Use PMETARECORD to create pointers that match the compiler memory model.

LPMOUSEHOOKSTRUCT

32-bit pointer to a MOUSEHOOKSTRUCT structure.

## LPMSG

32-bit pointer to an MSG structure. Use NPMSG to create 16-bit pointers. Use PMSG to create pointers that match the compiler memory model.

LPNCCALCSIZE\_PARAMS

32-bit pointer to an NCCALCSIZE\_PARAMS structure.

LPNEWCPLINFO

32-bit pointer to an NEWCPLINFO structure. Use PNEWCPLINFO to create pointers that match the compiler memory model.

## LPNEWTEXTMETRIC

32-bit pointer to a NEWTEXTMETRIC structure. Use NPNEWTEXTMETRIC to create 16-bit pointers. Use PNEWTEXTMETRIC to create pointers that match the compiler memory model.



LPOFSTRUCT

32-bit pointer to an OFSTRUCT structure. Use NPOFSTRUCT to create 16-bit pointers. Use POFSTRUCT to create pointers that match the compiler memory model.

LPOLECLIENT

32-bit pointer to OLECLIENT structure.

LPOLECLIENTVTBL

32-bit pointer to OLECLIENTVTBL structure.

LPOLEOBJECT

32-bit pointer to OLEOBJECT structure.

LPOLEOBJECTVTBL

32-bit pointer to OLEOBJECTVTBL structure.

LPOLESERVER

32-bit pointer to OLESERVER structure.

LPOLESERVERDOC

32-bit pointer to OLESERVERDOC structure.

LPOLESERVERDOCVTBL

32-bit pointer to OLESERVERDOCVTBL structure.



LPOLESERVERVTBL

32-bit pointer to OLESERVERVTBL structure.

LPOLESTREAM

32-bit pointer to OLESTREAM structure.

LPOLESTREAMVTBL

32-bit pointer to OLESTREAMVTBL structure.

LPOLETARGETDEVICE

32-bit pointer to OLETARGETDEVICE structure.

LPOPENFILENAME

32-bit pointer to OPENFILENAME structure.

LPOUTLINETEXTMETRIC

32-bit pointer to an OUTLINETEXTMETRIC structure.

LPPAINTSTRUCT

32-bit pointer to a PAINSTRUCT structure. Use NPPAINTSTRUCT to create 16-bit pointers. Use PPAINSTRUCT to create pointers that match the compiler memory model.

LPPALETTEENTRY

32-bit pointer to a PALETTEENTRY structure.



LPPOINT

32-bit pointer to a POINT structure. Use NPPOINT to create 16-bit pointers. Use PPOINT to create pointers that match the compiler memory model.

LPPOINTFX

32-bit pointer to a POINTFX structure.

LPPRINTDLG

32-bit pointer to a PRINTDLG structure.

LPRASTERIZER\_STATUS

32-bit pointer to a RASTERIZER\_STATUS structure.

LPRECT

32-bit pointer to a RECT structure. Use NPRECT to create 16-bit pointers. Use PRECT to create pointers that match the compiler memory model.

LPRGBQUAD

32-bit pointer to a RGBQUAD structure.

LPRGBTRIPLE

32-bit pointer to a RGBTRIPLE structure.

LPSEGINFO

32-bit pointer to a SEGINFO structure.



LPSIZE

32-bit pointer to a SIZE structure. Use NPSIZE to create 16-bit pointers. Use PSIZE to create pointers that match the compiler memory model.

LPSTR

32-bit pointer to a character string. Use NPSTR to create 16-bit pointers. Use PSTR to create pointers that match the compiler memory model.

LPTEXTMETRIC

32-bit pointer to a TEXTMETRIC structure. Use NPTEXTMETRIC to create 16-bit pointers. Use PTEXTMETRIC to create pointers that match the compiler memory model.

LPTTPOLYCURVE

32-bit pointer to a TTPOLYCURVE structure.

LPTTPOLYGONHEADER

32-bit pointer to a TTPOLYGONHEADER structure.

LPVOID

32-bit pointer to an unspecified type.

## LPWINDOWPLACEMENT

32-bit pointer to a WINDOWPLACEMENT structure. Use PWINDOWPLACEMENT to create pointers that match the compiler memory model.

LPWINDOWPOS

32-bit pointer to a WINDOWPOS structure.



LPWNDCLASS

32-bit pointer to a WNDCLASS structure. Use NPWNDCLASS to create 16-bit pointers. Use PWNDCLASS to create pointers that match the compiler memory model.

## LPWORD

32-bit pointer to a 16-bit unsigned value. Use PWORD to create pointers that match the compiler memory model.

## LRESULT

32-bit signed value returned from a window procedure or callback function.

MFENUMPROC

32-bit pointer to an EnumMetaFileProc callback function.

NEARPROC

16-bit pointer to a function.

## OLECLIPFORMAT

16-bit value used as a standard clipboard format.

## PATTERN

Equivalent to the LOGBRUSH structure. Use LPPATTERN to create 32-bit pointers. Use NPPATTERN to create 16-bit pointers. Use PPATTERN to create pointers that match the compiler memory model.

PCONVCONTEXT

32-bit pointer to a CONVCONTEXT structure.



PCONVINFO

32-bit pointer to a CONVINFO structure.

PHSZPAIR

32-bit pointer to a HSZPAIR structure.

PROPENUMPROC

32-bit pointer to an EnumPropFixedProc or EnumPropMovableProc callback function.

RSRCHDLRPROC

32-bit pointer to a LoadProc callback function.

UINT

16-bit unsigned value.

WNDENUMPROC

32-bit pointer to an EnumWindowsProc callback function.

WNDPROC

32-bit pointer to a window procedure.

WORD

16-bit unsigned value.



## WPARAM

16-bit signed value passed as a parameter to a window procedure or callback function.

## Button styles (3.1)

Value	Meaning
BS_3STATE	Creates a button that is the same as a check box, except that the box can be grayed (dimmed) as well as checked. The grayed state is used to show that the state of the check box is not determined.
BS_AUTO3STATE	Creates a button that is the same as a three-state check box, except that the box changes its state when the user selects it. The state cycles through checked, grayed, and normal.
BS_AUTOCHECKBOX	Creates a button that is the same as a check box, except that an X appears in the check box when the user selects the box; the X disappears (is cleared) the next time the user selects the box.
BS_AUTORADIOBUTTON	Creates a button that is the same as a radio button, except that when the user selects it, the button automatically highlights itself and clears (removes the selection from) any other buttons in the same group.
BS_CHECKBOX	Creates a small square that has text displayed to its right (unless this style is combined with the <u>BS_LEFTTEXT</u> style).
BS_DEFPUSHBUTTON	Creates a button that has a heavy black border. The user can select this button by pressing the ENTER key. This style is useful for enabling the user to quickly select the most likely option (the default option).
BS_GROUPBOX	Creates a rectangle in which other controls can be grouped. Any text associated with this style is displayed in the rectangle's upper-left corner.
BS_LEFTTEXT	Places text on the left side of the radio button or check box when combined with a radio button or check box style.
BS_OWNERDRAW	Creates an owner-drawn button. The owner window receives a <u>WM_MEASUREITEM</u> message when the button is created, and it receives a <u>WM_DRAWITEM</u> message when a visual aspect of the button has changed. The <u>BS_OWNERDRAW</u> style cannot be combined with any other button styles.
BS_PUSHBUTTON	Creates a push button that posts a <u>WM_COMMAND</u> message to the owner window when the user selects the button.
BS_RADIOBUTTON	Creates a small circle that has text displayed to its right (unless this style is combined with the <u>BS_LEFTTEXT</u> style). Radio buttons are usually used in groups of related but mutually exclusive choices.

### Combination box styles (3.1)

Style	Description
CBS_AUTOHSCROLL	Automatically scrolls the text in the edit control to the right when the user types a character at the end of the line. If this style is not set, only text that fits within the rectangular boundary is allowed.
CBS_DISABLENOSCROLL	Shows a disabled vertical scroll bar in the list box when the box does not contain enough items to scroll. Without this style, the scroll bar is hidden when the list box does not contain enough items.
CBS_DROPDOWN	Similar to <u>CBS_SIMPLE</u> , except that the list box is not displayed unless the user selects an icon next to the edit control.
CBS_DROPDOWNLIST	Similar to <u>CBS_DROPDOWN</u> , except that the edit control is replaced by a static text item that displays the current selection in the list box.
CBS_HASSTRINGS	Specifies that an owner-drawn combo box contains items consisting of strings. The combo box maintains the memory and pointers for the strings so the application can use the <u>CB_GETLBTEXT</u> message to retrieve the text for a particular item.
CBS_NOINTEGRALHEIGHT	Specifies that the size of the combo box is exactly the size specified by the application when it created the combo box. Normally, Windows sizes a combo box so that the combo box does not display partial items.
CBS_OEMCONVERT	Converts text entered in the combo-box edit control from the Windows character set to the OEM character set and then back to the Windows set. This ensures proper character conversion when the application calls the <u>AnsiToOem</u> function to convert a Windows string in the <u>combo box</u> to OEM characters. This style is most useful for combo boxes that contain filenames and applies only to combo boxes created with the <u>CBS_SIMPLE</u> or <u>CBS_DROPDOWN</u> styles.
CBS_OWNERDRAWFIXED	Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are all the same height. The owner window receives a <u>WM_MEASUREITEM</u> message when the combo box is created and a <u>WM_DRAWITEM</u> message when a visual aspect of the <u>combo box</u> has changed.
CBS_OWNERDRAWVARIABLE	Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are variable in height. The owner window receives a <u>WM_MEASUREITEM</u> message for each item in the <u>combo box</u> when the <u>combo box</u> is created and a <u>WM_DRAWITEM</u> message whenever the visual aspect of the <u>combo box</u> changes.
CBS_SIMPLE	Displays the list box at all times. The current selection in the list box is displayed in the edit control.
CBS_SORT	Automatically sorts strings entered into the list box.

### Edit control styles (3.1)

Style	Meaning
ES_AUTOHSCROLL	Automatically scrolls text to the right by 10 characters when the user types a character at the end of the line. When the user presses the ENTER key, the control scrolls all text back to position zero.
ES_AUTOVSCROLL	Automatically scrolls text up one page when the user presses ENTER on the last line.
ES_CENTER	Centers text in a multiline edit control.
ES_LEFT	Left aligns text.
ES_LOWERCASE	Converts all characters to lowercase as they are typed into the edit control.
ES_MULTILINE	Designates a multiline edit control. (The default is single-line edit control.) When the multiline edit control is in a dialog box, the default response to pressing the ENTER key is to activate the default button. To use the ENTER key as a carriage return, an application should use the <u>ES_WANTRETURN</u> style. When the multiline edit control is not in a dialog box and the <u>ES_AUTOVSCROLL</u> style is specified, the edit control shows as many lines as possible and scrolls vertically when the user presses the ENTER key. If <u>ES_AUTOVSCROLL</u> is not specified, the edit control shows as many lines as possible and beeps if the user presses ENTER when no more lines can be displayed. If the <u>ES_AUTOHSCROLL</u> style is specified, the multiline edit control automatically scrolls horizontally when the caret goes past the right edge of the control. To start a new line, the user must press ENTER. If <u>ES_AUTOHSCROLL</u> is not specified, the control automatically wraps words to the beginning of the next line when necessary. A new line is also started if the user presses ENTER. The position of the wordwrap is determined by the window size. If the window size changes, the wordwrap position changes and the text is redisplayed. Multiline edit controls can have scroll bars. An edit control with scroll bars processes its own scroll bar messages. Edit controls without scroll bars scroll as described in the previous two paragraphs and process any scroll messages sent by the parent window.
ES_NOHIDSEL	Negates the default behavior for an edit control. The default behavior is to hide the selection when the control loses the input focus and invert the selection when the control receives the input focus.
ES_OEMCONVERT	Converts text entered in the edit control from the Windows character set to the OEM character set and then back to the Windows set. This ensures proper character conversion when the application calls the <u>AnsiToOem</u> function to convert a Windows string in the edit control to OEM characters. This style is most useful for edit controls that contain filenames.
ES_PASSWORD	Displays all characters as an asterisk (*) as they are typed into the edit control. An application can use the <u>EM_SETPASSWORDCHAR</u> message to change the character that is displayed.
ES_READONLY	Prevents the user from typing or editing text in the edit control.
ES_RIGHT	Right aligns text in a multiline edit control.
ES_UPPERCASE	Converts all characters to uppercase as they are typed into the edit control.
ES_WANTRETURN	Specifies that a carriage return be inserted when the user presses the ENTER key while entering text into a multiline edit control in a dialog box. If this style is not specified, pressing the ENTER key has the same effect as pressing the dialog box's default push button. This style has no effect on a single-line edit control.

### List box styles (3.1)

Style	Meaning
LBS_DISABLENOSCROLL	Shows a disabled vertical scroll bar for the list box when the box does not contain enough items to scroll. If this style is not specified, the scroll bar is hidden when the list box does not contain enough items.
LBS_EXTENDEDSEL	Allows multiple items to be selected by using the SHIFT key and the mouse or special key combinations.
LBS_HASSTRINGS	Specifies that a list box contains items consisting of strings. The list box maintains the memory and pointers for the strings so the application can use the <u>LB_GETTEXT</u> message to retrieve the text for a particular item. By default, all list boxes except owner-drawn list boxes have this style. An application can create an owner-drawn list box either with or without this style.
LBS_MULTICOLUMN	Specifies a multicolumn list box that is scrolled horizontally. The <u>LB_SETCOLUMNWIDTH</u> message sets the width of the columns.
LBS_MULTIPLESEL	Turns string selection on or off each time the user clicks or double-clicks the string. Any number of strings can be selected.
LBS_NOINTEGRALHEIGHT	Specifies that the size of the list box is exactly the size specified by the application when it created the list box. Normally, Windows sizes a list box so that the list box does not display partial items.
LBS_NOREDRAW	Specifies that the list box's appearance is not updated when changes are made. This style can be changed at any time by sending a <u>WM_SETREDRAW</u> message.
LBS_NOTIFY	Notifies the parent window with an input message whenever the user clicks or double-clicks a string.
LBS_OWNERDRAWFIXED	Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are the same height. The owner window receives a <u>WM_MEASUREITEM</u> message when the list box is created and a <u>WM_DRAWITEM</u> message when a visual aspect of the list box has changed.
LBS_OWNERDRAWVARIABLE	Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are variable in height. The owner window receives a <u>WM_MEASUREITEM</u> message for each item in the combo box when the combo box is created and a <u>WM_DRAWITEM</u> message whenever the visual aspect of the <u>combo box</u> changes.
LBS_SORT	Sorts strings in the list box alphabetically.
LBS_STANDARD	Sorts strings in the list box alphabetically. The parent window receives an input message whenever the user clicks or double-clicks a string. The list box has borders on all sides.
LBS_USETABSTOPS	Allows a list box to recognize and expand tab characters when drawing its strings. The default tab positions are 32 dialog box units. (A dialog box unit is a horizontal or vertical distance. One horizontal dialog box unit is equal to one-fourth of the current dialog box base width unit. The dialog box base units are computed based on the height and width of the current system font. The <u>GetDialogBaseUnits</u> function returns the current dialog box base units in pixels.)
LBS_WANTKEYBOARDINPUT	Specifies that the owner of the list box receives <u>WM_VKEYTOITEM</u> or <u>WM_CHARTOITEM</u> messages whenever the user presses a key and the list box has the input focus. This allows an application to perform special processing on the keyboard input. If a list box has the

LBS\_HASSTRINGS style, the list box can receive WM\_VKEYTOITEM messages but not WM\_CHARTOITEM messages. If a list box does not have the LBS\_HASSTRINGS style, the list box can receive WM\_CHARTOITEM messages but not WM\_VKEYTOITEM messages.

### Scroll bar styles (3.1)

Style	Meaning
SBS_BOTTOMALIGN	Aligns the bottom edge of the scroll bar with the bottom edge of the rectangle defined by the following <u>CreateWindow</u> parameters: x, y, nWidth, and nHeight. The scroll bar has the default height for system scroll bars. Used with the SBS_HORZ style.
SBS_HORZ	Designates a horizontal scroll bar. If neither the SBS_BOTTOMALIGN nor SBS_TOPALIGN style is specified, the scroll bar has the height, width, and position specified by the <u>CreateWindow</u> parameters.
SBS_LEFTALIGN	Aligns the left edge of the scroll bar with the left edge of the rectangle defined by the <u>CreateWindow</u> parameters. The scroll bar has the default width for system scroll bars. Used with the SBS_VERT style.
SBS_RIGHTALIGN	Aligns the right edge of the scroll bar with the right edge of the rectangle defined by the <u>CreateWindow</u> parameters. The scroll bar has the default width for system scroll bars. Used with the SBS_VERT style.
SBS_SIZEBOX	Designates a size box. If neither the SBS_SIZEBOXBOTTOMRIGHTALIGN nor SBS_SIZEBOXTOPLEFTALIGN style is specified, the size box has the height, width, and position specified by the <u>CreateWindow</u> parameters.
SBS_SIZEBOXBOTTOMRIGHTALIGN	Aligns the lower-right corner of the size box with the lower-right corner of the rectangle specified by the <u>CreateWindow</u> parameters. The size box has the default size for system size boxes. Used with the SBS_SIZEBOX style.
SBS_SIZEBOXTOPLEFTALIGN	Aligns the upper-left corner of the size box with the upper-left corner of the rectangle specified by the following <u>CreateWindow</u> parameters: x, y, nWidth, and nHeight. The size box has the default size for system size boxes. Used with the SBS_SIZEBOX style.
SBS_TOPALIGN	Aligns the top edge of the scroll bar with the top edge of the rectangle defined by the <u>CreateWindow</u> parameters. The scroll bar has the default height for system scroll bars. Used with the SBS_HORZ style.
SBS_VERT	Designates a vertical scroll bar. If neither the SBS_RIGHTALIGN nor SBS_LEFTALIGN style is specified, the scroll bar has the height, width, and position specified by the <u>CreateWindow</u> parameters.

### Static control styles (3.1)

Style	Meaning
SS_BLACKFRAME	Specifies a box with a frame drawn in the same color as window frames. This color is black in the default Windows color scheme.
SS_BLACKRECT	Specifies a rectangle filled with the color used to draw window frames. This color is black in the default Windows color scheme.
SS_CENTER	Designates a simple rectangle and displays the given text centered in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next centered line.
SS_GRAYFRAME	Specifies a box with a frame drawn with the same color as the screen background (desktop). This color is gray in the default Windows color scheme.
SS_GRAYRECT	Specifies a rectangle filled with the color used to fill the screen background. This color is gray in the default Windows color scheme.
SS_ICON	Designates an icon displayed in the dialog box. The given text is the name of an icon (not a filename) defined elsewhere in the resource file. The nWidth and nHeight parameters are ignored; the icon automatically sizes itself.
SS_LEFT	Designates a simple rectangle and displays the given text left-aligned in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next left-aligned line.
SS_LEFTNOWORDWRAP	Designates a simple rectangle and displays the given text left-aligned in the rectangle. Tabs are expanded but words are not wrapped. Text that extends past the end of a line is clipped.
SS_NOPREFIX	Prevents interpretation of any & characters in the control's text as accelerator prefix characters (which are displayed with the & removed and the next character in the string underlined). This static control style may be included with any of the defined static controls.  You can combine <u>SS_NOPREFIX</u> with other styles by using the bitwise OR operator. This is most often used when filenames or other strings that may contain an & need to be displayed in a static control in a dialog box.
SS_RIGHT	Designates a simple rectangle and displays the given text right-aligned in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next right-aligned line.
SS_SIMPLE	Designates a simple rectangle and displays a single line of text left-aligned in the rectangle. The line of text cannot be shortened or altered in any way. (The control's parent window or dialog box must not process the <u>WM_CTLCOLOR</u> message.)
SS_WHITEFRAME	Specifies a box with a frame drawn in the same color as window backgrounds. This color is white in the default Windows color scheme.
SS_WHITERECT	Specifies a rectangle filled with the color used to fill window backgrounds. This color is white in the default Windows color scheme.



## BS\_3STATE

Creates a button that is the same as a check box, except that the box can be grayed (dimmed) as well as checked. The grayed state is used to show that the state of the check box is not determined.

## BS\_AUTO3STATE

Creates a button that is the same as a three-state check box, except that the box changes its state when the user selects it. The state cycles through checked, grayed, and normal.

## BS\_AUTOCHECKBOX

Creates a button that is the same as a check box, except that an X appears in the check box when the user selects the box; the X disappears (is cleared) the next time the user selects the box.

## BS\_AUTORADIOBUTTON

Creates a button that is the same as a radio button, except that when the user selects it, the button automatically highlights itself and clears (removes the selection from) any other buttons in the same group.

**BS\_CHECKBOX**

Creates a small square that has text displayed to its right (unless this style is combined with the BS\_LEFTTEXT style).

## BS\_DEFPUSHBUTTON

Creates a button that has a heavy black border. The user can select this button by pressing the ENTER key. This style is useful for enabling the user to quickly select the most likely option (the default option).

## BS\_GROUPBOX

Creates a rectangle in which other controls can be grouped. Any text associated with this style is displayed in the rectangle's upper-left corner.

**BS\_LEFTTEXT**

Places text on the left side of the radio button or check box when combined with a radio button or check box style.



## BS\_OWNERDRAW

Creates an owner-drawn button. The owner window receives a WM\_MEASUREITEM message when the button is created, and it receives a WM\_DRAWITEM message when a visual aspect of the button has changed. The BS\_OWNERDRAW style cannot be combined with any other button styles.

## BS\_PUSHBUTTON

Creates a push button that posts a WM\_COMMAND message to the owner window when the user selects the button.

## BS\_RADIOBUTTON

Creates a small circle that has text displayed to its right (unless this style is combined with the BS\_LEFTTEXT style). Radio buttons are usually used in groups of related but mutually exclusive choices.

## CBS\_AUTOHSCROLL

Automatically scrolls the text in the edit control to the right when the user types a character at the end of the line. If this style is not set, only text that fits within the rectangular boundary is allowed.

## CBS\_DISABLENOSCROLL

Shows a disabled vertical scroll bar in the list box when the box does not contain enough items to scroll. Without this style, the scroll bar is hidden when the list box does not contain enough items.

## CBS\_DROPDOWN

Similar to CBS\_SIMPLE, except that the list box is not displayed unless the user selects an icon next to the edit control.

## CBS\_DROPDOWNLIST

Similar to CBS\_DROPDOWN, except that the edit control is replaced by a static text item that displays the current selection in the list box.

## CBS\_HASSTRINGS

Specifies that an owner-drawn combo box contains items consisting of strings. The combo box maintains the memory and pointers for the strings so the application can use the CB\_GETLBTEXT message to retrieve the text for a particular item.



### CBS\_NOINTEGRALHEIGHT

Specifies that the size of the combo box is exactly the size specified by the application when it created the combo box. Normally, Windows sizes a combo box so that the combo box does not display partial items.

## CBS\_OEMCONVERT

Converts text entered in the combo-box edit control from the Windows character set to the OEM character set and then back to the Windows set. This ensures proper character conversion when the application calls the AnsiToOem function to convert a Windows string in the combo box to OEM characters. This style is most useful for combo boxes that contain filenames and applies only to combo boxes created with the CBS\_SIMPLE or CBS\_DROPDOWN styles.

### CBS\_OWNERDRAWFIXED

Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are all the same height. The owner window receives a WM\_MEASUREITEM message when the combo box is created and a WM\_DRAWITEM message when a visual aspect of the combo box has changed.

### CBS\_OWNERDRAWVARIABLE

Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are variable in height. The owner window receives a WM\_MEASUREITEM message for each item in the combo box when the combo box is created and a WM\_DRAWITEM message whenever the visual aspect of the combo box changes.

CBS\_SIMPLE

Displays the list box at all times. The current selection in the list box is displayed in the edit control.

CBS\_SORT

Automatically sorts strings entered into the list box.

## ES\_AUTOHSCROLL

Automatically scrolls text to the right by 10 characters when the user types a character at the end of the line. When the user presses the ENTER key, the control scrolls all text back to position zero.

ES\_AUTOVSCROLL

Automatically scrolls text up one page when the user presses ENTER on the last line.



ES\_CENTER

Centers text in a multiline edit control.

ES\_LEFT

Left aligns text.

ES\_LOWERCASE

Converts all characters to lowercase as they are typed into the edit control.

ES\_MULTILINE

Designates a multiline edit control. (The default is single-line edit control.)

When

the multiline edit control is in a dialog box, the default response to pressing the ENTER key is to activate the default button. To use the ENTER key as a carriage return, an application should use the `ES_WANTRETURN` style. When the multiline edit control is not in a dialog box and the `ES_AUTOVSCROLL` style is specified, the edit control shows as many lines as possible and scrolls vertically when the user presses the ENTER key. If `ES_AUTOVSCROLL` is not specified, the edit control shows as many lines as possible and beeps if the user presses ENTER when no more lines can be displayed.

If

the `ES_AUTOHSCROLL` style is specified, the multiline edit control automatically scrolls horizontally when the caret goes past the right edge of the control. To start a new line, the user must press `ENTER`. If `ES_AUTOHSCROLL` is not specified, the control automatically wraps words to the beginning of the next line when necessary. A new line is also started if the user presses `ENTER`. The position of the wordwrap is determined by the window size. If the window size changes, the wordwrap position changes and the text is redisplayed. Multiline edit controls can have scroll bars. An edit control with scroll bars processes its own scroll bar messages. Edit controls without scroll bars scroll as described in the previous two paragraphs and process any scroll messages sent by the parent window.

ES\_NOHIDSEL

Negates the default behavior for an edit control. The default behavior is to hide the selection when the control loses the input focus and invert the selection when the control receives the input focus.

## ES\_OEMCONVERT

Converts text entered in the edit control from the Windows character set to the OEM character set and then back to the Windows set. This ensures proper character conversion when the application calls the AnsiToOem function to convert a Windows string in the edit control to OEM characters. This style is most useful for edit controls that contain filenames.



## ES\_PASSWORD

Displays all characters as an asterisk (\*) as they are typed into the edit control. An application can use the EM\_SETPASSWORDCHAR message to change the character that is displayed.

ES\_READONLY

Prevents the user from typing or editing text in the edit control.

ES\_RIGHT

Right aligns text in a multiline edit control.

ES\_UPPERCASE

Converts all characters to uppercase as they are typed into the edit control.

## ES\_WANTRETURN

Specifies that a carriage return be inserted when the user presses the ENTER key while entering text into a multiline edit control in a dialog box. If this style is not specified, pressing the ENTER key has the same effect as pressing the dialog box's default push button. This style has no effect on a single-line edit control.

## LBS\_DISABLENOSCROLL

Shows a disabled vertical scroll bar for the list box when the box does not contain enough items to scroll. If this style is not specified, the scroll bar is hidden when the list box does not contain enough items.

LBS\_EXTENDESEL

Allows multiple items to be selected by using the SHIFT key and the mouse or special key combinations.

## LBS\_HASSTRINGS

Specifies that a list box contains items consisting of strings. The list box maintains the memory and pointers for the strings so the application can use the LB\_GETTEXT message to retrieve the text for a particular item. By default, all list boxes except owner-drawn list boxes have this style. An application can create an owner-drawn list box either with or without this style.



## LBS\_MULTICOLUMN

Specifies a multicolumn list box that is scrolled horizontally. The LB\_SETCOLUMNWIDTH message sets the width of the columns.

## LBS\_MULTIPLESEL

Turns string selection on or off each time the user clicks or double-clicks the string. Any number of strings can be selected.

## LBS\_NOINTEGRALHEIGHT

Specifies that the size of the list box is exactly the size specified by the application when it created the list box. Normally, Windows sizes a list box so that the list box does not display partial items.

LBS\_NOREDRAW

Specifies that the list box's appearance is not updated when changes are made. This style can be changed at any time by sending a WM\_SETREDRAW message.

LBS\_NOTIFY

Notifies the parent window with an input message whenever the user clicks or double-clicks a string.

## LBS\_OWNERDRAWFIXED

Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are the same height. The owner window receives a WM\_MEASUREITEM message when the list box is created and a WM\_DRAWITEM message when a visual aspect of the list box has changed.

## LBS\_OWNERDRAWVARIABLE

Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are variable in height. The owner window receives a WM\_MEASUREITEM message for each item in the combo box when the combo box is created and a WM\_DRAWITEM message whenever the visual aspect of the combo box changes.

LBS\_SORT

Sorts strings in the list box alphabetically.



## LBS\_STANDARD

Sorts strings in the list box alphabetically. The parent window receives an input message whenever the user clicks or double-clicks a string. The list box has borders on all sides.

## LBS\_USETABSTOPS

Allows a list box to recognize and expand tab characters when drawing its strings. The default tab positions are 32 dialog box units. (A dialog box unit is a horizontal or vertical distance. One horizontal dialog box unit is equal to one-fourth of the current dialog box base width unit. The dialog box base units are computed based on the height and width of the current system font. The GetDialogBaseUnits function returns the current dialog box base units in pixels.)

## LBS\_WANTKEYBOARDINPUT

Specifies that the owner of the list box receives WM\_VKEYTOITEM or WM\_CHARTOITEM messages whenever the user presses a key and the list box has the input focus. This allows an application to perform special processing on the keyboard input. If a list box has the LBS\_HASSTRINGS style, the list box can receive WM\_VKEYTOITEM messages but not WM\_CHARTOITEM messages. If a list box does not have the LBS\_HASSTRINGS style, the list box can receive WM\_CHARTOITEM messages but not WM\_VKEYTOITEM messages.

## SBS\_BOTTOMALIGN

Aligns the bottom edge of the scroll bar with the bottom edge of the rectangle defined by the following CreateWindow parameters: *x*, *y*, *nWidth*, and *nHeight*. The scroll bar has the default height for system scroll bars. Used with the SBS\_HORZ style.

## SBS\_HORZ

Designates a horizontal scroll bar. If neither the SBS\_BOTTOMALIGN nor SBS\_TOPALIGN style is specified, the scroll bar has the height, width, and position specified by the CreateWindow parameters.

## SBS\_LEFTALIGN

Aligns the left edge of the scroll bar with the left edge of the rectangle defined by the `CreateWindow` parameters. The scroll bar has the default width for system scroll bars. Used with the SBS\_VERT style.

## SBS\_RIGHTALIGN

Aligns the right edge of the scroll bar with the right edge of the rectangle defined by the `CreateWindow` parameters. The scroll bar has the default width for system scroll bars. Used with the SBS\_VERT style.

## SBS\_SIZEBOX

Designates a size box. If neither the SBS\_SIZEBOXBOTTOMRIGHTALIGN nor SBS\_SIZEBOXTOPLEFTALIGN style is specified, the size box has the height, width, and position specified by the CreateWindow parameters.



#### SBS\_SIZEBOXBOTTOMRIGHTALIGN

Aligns the lower-right corner of the size box with the lower-right corner of the rectangle specified by the `CreateWindow` parameters. The size box has the default size for system size boxes. Used with the SBS\_SIZEBOX style.

## SBS\_SIZEBOXTOPLEFTALIGN

Aligns the upper-left corner of the size box with the upper-left corner of the rectangle specified by the following CreateWindow parameters: x, y, nWidth, and nHeight. The size box has the default size for system size boxes. Used with the SBS\_SIZEBOX style.

## SBS\_TOPALIGN

Aligns the top edge of the scroll bar with the top edge of the rectangle defined by the `CreateWindow` parameters. The scroll bar has the default height for system scroll bars. Used with the SBS\_HORZ style.

## SBS\_VERT

Designates a vertical scroll bar. If neither the SBS\_RIGHTALIGN nor SBS\_LEFTALIGN style is specified, the scroll bar has the height, width, and position specified by the CreateWindow parameters.

## SS\_BLACKFRAME

Specifies a box with a frame drawn in the same color as window frames. This color is black in the default Windows color scheme.

**SS\_BLACKRECT**

Specifies a rectangle filled with the color used to draw window frames. This color is black in the default Windows color scheme.

## SS\_CENTER

Designates a simple rectangle and displays the given text centered in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next centered line.

## SS\_GRAYFRAME

Specifies a box with a frame drawn with the same color as the screen background (desktop). This color is gray in the default Windows color scheme.



## SS\_GRAYRECT

Specifies a rectangle filled with the color used to fill the screen background. This color is gray in the default Windows color scheme.

## SS\_ICON

Designates an icon displayed in the dialog box. The given text is the name of an icon (not a filename) defined elsewhere in the resource file. The nWidth and nHeight parameters are ignored; the icon automatically sizes itself.

## SS\_LEFT

Designates a simple rectangle and displays the given text left-aligned in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next left-aligned line.

## SS\_LEFTNOWORDWRAP

Designates a simple rectangle and displays the given text left-aligned in the rectangle. Tabs are expanded but words are not wrapped. Text that extends past the end of a line is clipped.

## SS\_NOPREFIX

Prevents interpretation of any & characters in the control's text as accelerator prefix characters (which are displayed with the & removed and the next character in the string underlined). This static control style may be included with any of the defined static controls. You can combine SS\_NOPREFIX with other styles by using the bitwise OR operator. This is most often used when filenames or other strings that may contain an & need to be displayed in a static control in a dialog box.

## SS\_RIGHT

Designates a simple rectangle and displays the given text right-aligned in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next right-aligned line.

## SS\_SIMPLE

Designates a simple rectangle and displays a single line of text left-aligned in the rectangle. The line of text cannot be shortened or altered in any way. (The control's parent window or dialog box must not process the WM\_CTLCOLOR message.)

## SS\_WHITEFRAME

Specifies a box with a frame drawn in the same color as window backgrounds. This color is white in the default Windows color scheme.



## SS\_WHITERECT

Specifies a rectangle filled with the color used to fill window backgrounds. This color is white in the default Windows color scheme.

## Module and Library Names

This topic lists the module and import libraries associated with each Microsoft Windows function.

Function	Module	Import library
<u>AbortDoc</u>	GDI	LIBW.LIB
<u>AccessResource</u>	KERNEL	LIBW.LIB
<u>AddAtom</u>	KERNEL	LIBW.LIB
<u>AddFontResource</u>	GDI	LIBW.LIB
<u>AdjustWindowRect</u>	USER	LIBW.LIB
<u>AdjustWindowRectEx</u>	USER	LIBW.LIB
<u>AllocDiskSpace</u>	STRESS	STRESS.LIB
<u>AllocDStoCSAlias</u>	KERNEL	LIBW.LIB
<u>AllocFileHandles</u>	STRESS	STRESS.LIB
<u>AllocGDI Mem</u>	STRESS	STRESS.LIB
<u>AllocMem</u>	STRESS	STRESS.LIB
<u>AllocResource</u>	KERNEL	LIBW.LIB
<u>AllocSelector</u>	KERNEL	LIBW.LIB
<u>AllocUserMem</u>	STRESS	STRESS.LIB
<u>AnimatePalette</u>	GDI	LIBW.LIB
<u>AnsiLower</u>	USER	LIBW.LIB
<u>AnsiLowerBuff</u>	USER	LIBW.LIB
<u>AnsiNext</u>	USER	LIBW.LIB
<u>AnsiPrev</u>	USER	LIBW.LIB
<u>AnsiToOem</u>	KEYBOARD	LIBW.LIB
<u>AnsiToOemBuff</u>	KEYBOARD	LIBW.LIB
<u>AnsiUpper</u>	USER	LIBW.LIB
<u>AnsiUpperBuff</u>	USER	LIBW.LIB
<u>AnyPopup</u>	USER	LIBW.LIB
<u>AppendMenu</u>	USER	LIBW.LIB
<u>Arc</u>	GDI	LIBW.LIB
<u>ArrangeIconicWindows</u>	USER	LIBW.LIB
<u>BeginDeferWindowPos</u>	USER	LIBW.LIB
<u>BeginPaint</u>	USER	LIBW.LIB
<u>BitBlt</u>	GDI	LIBW.LIB
<u>BringWindowToTop</u>	USER	LIBW.LIB
<u>BuildCommDCB</u>	USER	LIBW.LIB
<u>CallMsgFilter</u>	USER	LIBW.LIB
<u>CallNextHookEx</u>	USER	LIBW.LIB
<u>CallWindowProc</u>	USER	LIBW.LIB
<u>Catch</u>	KERNEL	LIBW.LIB
<u>ChangeClipboardChain</u>	USER	LIBW.LIB
<u>ChangeMenu</u>	USER	LIBW.LIB
<u>CheckDlgButton</u>	USER	LIBW.LIB
<u>CheckMenuItem</u>	USER	LIBW.LIB
<u>CheckRadioButton</u>	USER	LIBW.LIB
<u>ChildWindowFromPoint</u>	USER	LIBW.LIB
<u>ChooseColor</u>	COMMDLG	COMMDLG.LIB
<u>ChooseFont</u>	COMMDLG	COMMDLG.LIB
<u>Chord</u>	GDI	LIBW.LIB
<u>ClassFirst</u>	TOOLHELP	TOOLHELP.LIB
<u>ClassNext</u>	TOOLHELP	TOOLHELP.LIB
<u>ClearCommBreak</u>	USER	LIBW.LIB

<u>ClientToScreen</u>	USER	LIBW.LIB
<u>ClipCursor</u>	USER	LIBW.LIB
<u>CloseClipboard</u>	USER	LIBW.LIB
<u>CloseComm</u>	USER	LIBW.LIB
<u>CloseDriver</u>	USER	LIBW.LIB
<u>CloseMetaFile</u>	GDI	LIBW.LIB
<u>CloseWindow</u>	USER	LIBW.LIB
<u>CombineRgn</u>	GDI	LIBW.LIB
<u>CommDlgExtendedError</u>	COMMDLG	COMMDLG.LIB
<u>CopyCursor</u>	USER	LIBW.LIB
<u>CopyIcon</u>	USER	LIBW.LIB
<u>CopyLZFile</u>	LZEXPAND	LZEXPAND.LIB
<u>CopyMetaFile</u>	GDI	LIBW.LIB
<u>CopyRect</u>	USER	LIBW.LIB
<u>CountClipboardFormats</u>	USER	LIBW.LIB
<u>CreateBitmap</u>	GDI	LIBW.LIB
<u>CreateBitmapIndirect</u>	GDI	LIBW.LIB
<u>CreateBrushIndirect</u>	GDI	LIBW.LIB
<u>CreateCaret</u>	USER	LIBW.LIB
<u>CreateCompatibleBitmap</u>	GDI	LIBW.LIB
<u>CreateCompatibleDC</u>	GDI	LIBW.LIB
<u>CreateCursor</u>	USER	LIBW.LIB
<u>CreateDC</u>	GDI	LIBW.LIB
<u>CreateDialog</u>	USER	LIBW.LIB
<u>CreateDialogIndirect</u>	USER	LIBW.LIB
<u>CreateDialogIndirectParam</u>	USER	LIBW.LIB
<u>CreateDialogParam</u>	USER	LIBW.LIB
<u>CreateDIBitmap</u>	GDI	LIBW.LIB
<u>CreateDIBPatternBrush</u>	GDI	LIBW.LIB
<u>CreateDiscardableBitmap</u>	GDI	LIBW.LIB
<u>CreateEllipticRgn</u>	GDI	LIBW.LIB
<u>CreateEllipticRgnIndirect</u>	GDI	LIBW.LIB
<u>CreateFont</u>	GDI	LIBW.LIB
<u>CreateFontIndirect</u>	GDI	LIBW.LIB
<u>CreateHatchBrush</u>	GDI	LIBW.LIB
<u>CreateIC</u>	GDI	LIBW.LIB
<u>CreateIcon</u>	USER	LIBW.LIB
<u>CreateMenu</u>	USER	LIBW.LIB
<u>CreateMetaFile</u>	GDI	LIBW.LIB
<u>CreatePalette</u>	GDI	LIBW.LIB
<u>CreatePatternBrush</u>	GDI	LIBW.LIB
<u>CreatePen</u>	GDI	LIBW.LIB
<u>CreatePenIndirect</u>	GDI	LIBW.LIB
<u>CreatePolygonRgn</u>	GDI	LIBW.LIB
<u>CreatePolyPolygonRgn</u>	GDI	LIBW.LIB
<u>CreatePopupMenu</u>	USER	LIBW.LIB
<u>CreateRectRgn</u>	GDI	LIBW.LIB
<u>CreateRectRgnIndirect</u>	GDI	LIBW.LIB
<u>CreateRoundRectRgn</u>	GDI	LIBW.LIB
<u>CreateScalableFontResource</u>	GDI	LIBW.LIB
<u>CreateSolidBrush</u>	GDI	LIBW.LIB
<u>CreateWindow</u>	USER	LIBW.LIB

<u>CreateWindowEx</u>	USER	LIBW.LIB
<u>DdeAbandonTransaction</u>	DDEML	DDEML.LIB
<u>DdeAccessData</u>	DDEML	DDEML.LIB
<u>DdeAddData</u>	DDEML	DDEML.LIB
<u>DdeClientTransaction</u>	DDEML	DDEML.LIB
<u>DdeCmpStringHandles</u>	DDEML	DDEML.LIB
<u>DdeConnect</u>	DDEML	DDEML.LIB
<u>DdeConnectList</u>	DDEML	DDEML.LIB
<u>DdeCreateDataHandle</u>	DDEML	DDEML.LIB
<u>DdeCreateStringHandle</u>	DDEML	DDEML.LIB
<u>DdeDisconnect</u>	DDEML	DDEML.LIB
<u>DdeDisconnectList</u>	DDEML	DDEML.LIB
<u>DdeEnableCallback</u>	DDEML	DDEML.LIB
<u>DdeFreeDataHandle</u>	DDEML	DDEML.LIB
<u>DdeFreeStringHandle</u>	DDEML	DDEML.LIB
<u>DdeGetData</u>	DDEML	DDEML.LIB
<u>DdeGetLastError</u>	DDEML	DDEML.LIB
<u>DdeInitialize</u>	DDEML	DDEML.LIB
<u>DdeKeepStringHandle</u>	DDEML	DDEML.LIB
<u>DdeNameService</u>	DDEML	DDEML.LIB
<u>DdePostAdvise</u>	DDEML	DDEML.LIB
<u>DdeQueryConvInfo</u>	DDEML	DDEML.LIB
<u>DdeQueryNextServer</u>	DDEML	DDEML.LIB
<u>DdeQueryString</u>	DDEML	DDEML.LIB
<u>DdeReconnect</u>	DDEML	DDEML.LIB
<u>DdeSetUserHandle</u>	DDEML	DDEML.LIB
<u>DdeUnaccessData</u>	DDEML	DDEML.LIB
<u>DdeUninitialize</u>	DDEML	DDEML.LIB
<u>DebugBreak</u>	KERNEL	LIBW.LIB
<u>DebugOutput</u>	KERNEL	LIBW.LIB
<u>DefDlgProc</u>	USER	LIBW.LIB
<u>DefDriverProc</u>	USER	LIBW.LIB
<u>DeferWindowPos</u>	USER	LIBW.LIB
<u>DefFrameProc</u>	USER	LIBW.LIB
<u>DefHookProc</u>	USER	LIBW.LIB
<u>DefMDIChildProc</u>	USER	LIBW.LIB
<u>DefScreenSaverProc</u>	—	SCRNSAVE.LIB
<u>DefWindowProc</u>	USER	LIBW.LIB
<u>DeleteAtom</u>	KERNEL	LIBW.LIB
<u>DeleteDC</u>	GDI	LIBW.LIB
<u>DeleteMenu</u>	USER	LIBW.LIB
<u>DeleteMetaFile</u>	GDI	LIBW.LIB
<u>DeleteObject</u>	GDI	LIBW.LIB
<u>DestroyCaret</u>	USER	LIBW.LIB
<u>DestroyCursor</u>	USER	LIBW.LIB
<u>DestroyIcon</u>	USER	LIBW.LIB
<u>DestroyMenu</u>	USER	LIBW.LIB
<u>DestroyWindow</u>	USER	LIBW.LIB
<u>DialogBox</u>	USER	LIBW.LIB
<u>DialogBoxIndirect</u>	USER	LIBW.LIB
<u>DialogBoxIndirectParam</u>	USER	LIBW.LIB
<u>DialogBoxParam</u>	USER	LIBW.LIB

<u>DirectedYield</u>	KERNEL	LIBW.LIB
<u>DispatchMessage</u>	USER	LIBW.LIB
<u>DlgChangePassword</u>	—	SCRNSAVE.LIB
<u>DlgDirList</u>	USER	LIBW.LIB
<u>DlgDirListComboBox</u>	USER	LIBW.LIB
<u>DlgDirSelect</u>	USER	LIBW.LIB
<u>DlgDirSelectComboBox</u>	USER	LIBW.LIB
<u>DlgDirSelectComboBoxEx</u>	USER	LIBW.LIB
<u>DlgDirSelectEx</u>	USER	LIBW.LIB
<u>DlgGetPassword</u>	—	SCRNSAVE.LIB
<u>DlgInvalidPassword</u>	—	SCRNSAVE.LIB
<u>DOS3Call</u>	KERNEL	LIBW.LIB
<u>DPtoLP</u>	GDI	LIBW.LIB
<u>DragAcceptFiles</u>	SHELL	SHELL.LIB
<u>DragFinish</u>	SHELL	SHELL.LIB
<u>DragQueryFile</u>	SHELL	SHELL.LIB
<u>DragQueryPoint</u>	SHELL	SHELL.LIB
<u>DrawFocusRect</u>	USER	LIBW.LIB
<u>DrawIcon</u>	USER	LIBW.LIB
<u>DrawMenuBar</u>	USER	LIBW.LIB
<u>DrawText</u>	USER	LIBW.LIB
<u>Ellipse</u>	GDI	LIBW.LIB
<u>EmptyClipboard</u>	USER	LIBW.LIB
<u>EnableCommNotification</u>	USER	LIBW.LIB
<u>EnableHardwareInput</u>	USER	LIBW.LIB
<u>EnableMenuItem</u>	USER	LIBW.LIB
<u>EnableScrollBar</u>	USER	LIBW.LIB
<u>EnableWindow</u>	USER	LIBW.LIB
<u>EndDeferWindowPos</u>	USER	LIBW.LIB
<u>EndDialog</u>	USER	LIBW.LIB
<u>EndDoc</u>	GDI	LIBW.LIB
<u>EndPage</u>	GDI	LIBW.LIB
<u>EndPaint</u>	USER	LIBW.LIB
<u>EnumChildWindows</u>	USER	LIBW.LIB
<u>EnumClipboardFormats</u>	USER	LIBW.LIB
<u>EnumFontFamilies</u>	GDI	LIBW.LIB
<u>EnumFonts</u>	GDI	LIBW.LIB
<u>EnumMetaFile</u>	GDI	LIBW.LIB
<u>EnumObjects</u>	GDI	LIBW.LIB
<u>EnumProps</u>	USER	LIBW.LIB
<u>EnumTaskWindows</u>	USER	LIBW.LIB
<u>EnumWindows</u>	USER	LIBW.LIB
<u>EqualRect</u>	USER	LIBW.LIB
<u>EqualRgn</u>	GDI	LIBW.LIB
<u>Escape</u>	GDI	LIBW.LIB
<u>EscapeCommFunction</u>	USER	LIBW.LIB
<u>ExcludeClipRect</u>	GDI	LIBW.LIB
<u>ExcludeUpdateRgn</u>	USER	LIBW.LIB
<u>ExitWindows</u>	USER	LIBW.LIB
<u>ExitWindowsExec</u>	USER	LIBW.LIB
<u>ExtFloodFill</u>	GDI	LIBW.LIB
<u>ExtractIcon</u>	SHELL	SHELL.LIB

<u>ExtTextOut</u>	GDI	LIBW.LIB
<u>FatalAppExit</u>	KERNEL	LIBW.LIB
<u>FatalExit</u>	KERNEL	LIBW.LIB
<u>FillRect</u>	USER	LIBW.LIB
<u>FillRgn</u>	GDI	LIBW.LIB
<u>FindAtom</u>	KERNEL	LIBW.LIB
<u>FindExecutable</u>	SHELL	SHELL.LIB
<u>FindResource</u>	KERNEL	LIBW.LIB
<u>FindText</u>	COMMDLG	COMMDLG.LIB
<u>FindWindow</u>	USER	LIBW.LIB
<u>FlashWindow</u>	USER	LIBW.LIB
<u>FloodFill</u>	GDI	LIBW.LIB
<u>FlushComm</u>	USER	LIBW.LIB
<u>FrameRect</u>	USER	LIBW.LIB
<u>FrameRgn</u>	GDI	LIBW.LIB
<u>FreeAllGDI Mem</u>	STRESS	STRESS.LIB
<u>FreeAllMem</u>	STRESS	STRESS.LIB
<u>FreeAllUserMem</u>	STRESS	STRESS.LIB
<u>FreeLibrary</u>	KERNEL	LIBW.LIB
<u>FreeModule</u>	KERNEL	LIBW.LIB
<u>FreeProcInstance</u>	KERNEL	LIBW.LIB
<u>FreeResource</u>	KERNEL	LIBW.LIB
<u>FreeSelector</u>	KERNEL	LIBW.LIB
<u>GetActiveWindow</u>	USER	LIBW.LIB
<u>GetAspectRatioFilter</u>	GDI	LIBW.LIB
<u>GetAspectRatioFilterEx</u>	GDI	LIBW.LIB
<u>GetAsyncKeyState</u>	USER	LIBW.LIB
<u>GetAtomHandle</u>	KERNEL	LIBW.LIB
<u>GetAtomName</u>	KERNEL	LIBW.LIB
<u>GetBitmapBits</u>	GDI	LIBW.LIB
<u>GetBitmapDimension</u>	GDI	LIBW.LIB
<u>GetBitmapDimensionEx</u>	GDI	LIBW.LIB
<u>GetBkColor</u>	GDI	LIBW.LIB
<u>GetBkMode</u>	GDI	LIBW.LIB
<u>GetBoundsRect</u>	GDI	LIBW.LIB
<u>GetBrushOrg</u>	GDI	LIBW.LIB
<u>GetBrushOrgEx</u>	GDI	LIBW.LIB
<u>GetCapture</u>	USER	LIBW.LIB
<u>GetCaretBlinkTime</u>	USER	LIBW.LIB
<u>GetCaretPos</u>	USER	LIBW.LIB
<u>GetCharABCWidths</u>	GDI	LIBW.LIB
<u>GetCharWidth</u>	GDI	LIBW.LIB
<u>GetClassInfo</u>	USER	LIBW.LIB
<u>GetClassLong</u>	USER	LIBW.LIB
<u>GetClassName</u>	USER	LIBW.LIB
<u>GetClassWord</u>	USER	LIBW.LIB
<u>GetClientRect</u>	USER	LIBW.LIB
<u>GetClipboardData</u>	USER	LIBW.LIB
<u>GetClipboardFormatName</u>	USER	LIBW.LIB
<u>GetClipboardOwner</u>	USER	LIBW.LIB
<u>GetClipboardViewer</u>	USER	LIBW.LIB
<u>GetClipBox</u>	GDI	LIBW.LIB

<u>GetClipCursor</u>	USER	LIBW.LIB
<u>GetCodeHandle</u>	KERNEL	LIBW.LIB
<u>GetCodeInfo</u>	KERNEL	LIBW.LIB
<u>GetCommError</u>	USER	LIBW.LIB
<u>GetCommEventMask</u>	USER	LIBW.LIB
<u>GetCommState</u>	USER	LIBW.LIB
<u>GetCurrentPDB</u>	KERNEL	LIBW.LIB
<u>GetCurrentPosition</u>	GDI	LIBW.LIB
<u>GetCurrentPositionEx</u>	GDI	LIBW.LIB
<u>GetCurrentTask</u>	KERNEL	LIBW.LIB
<u>GetCurrentTime</u>	USER	LIBW.LIB
<u>GetCursor</u>	USER	LIBW.LIB
<u>GetCursorPos</u>	USER	LIBW.LIB
<u>GetDC</u>	USER	LIBW.LIB
<u>GetDCEx</u>	USER	LIBW.LIB
<u>GetDCOrg</u>	GDI	LIBW.LIB
<u>GetDesktopWindow</u>	USER	LIBW.LIB
<u>GetDeviceCaps</u>	GDI	LIBW.LIB
<u>GetDialogBaseUnits</u>	USER	LIBW.LIB
<u>GetDIBits</u>	GDI	LIBW.LIB
<u>GetDlgCtrlID</u>	USER	LIBW.LIB
<u>GetDlgItem</u>	USER	LIBW.LIB
<u>GetDlgItemInt</u>	USER	LIBW.LIB
<u>GetDlgItemText</u>	USER	LIBW.LIB
<u>GetDOSEnvironment</u>	KERNEL	LIBW.LIB
<u>GetDoubleClickTime</u>	USER	LIBW.LIB
<u>GetDriverInfo</u>	USER	LIBW.LIB
<u>GetDriverModuleHandle</u>	USER	LIBW.LIB
<u>GetDriveType</u>	KERNEL	LIBW.LIB
<u>GetExpandedName</u>	LZEXPAND	LZEXPAND.LIB
<u>GetFileResource</u>	VER	VER.LIB
<u>GetFileResourceSize</u>	VER	VER.LIB
<u>GetFileTitle</u>	COMMDLG	COMMDLG.LIB
<u>GetFileVersionInfo</u>	VER	VER.LIB
<u>GetFileVersionInfoSize</u>	VER	VER.LIB
<u>GetFocus</u>	USER	LIBW.LIB
<u>GetFontData</u>	GDI	LIBW.LIB
<u>GetFreeFileHandles</u>	STRESS	STRESS.LIB
<u>GetFreeSpace</u>	KERNEL	LIBW.LIB
<u>GetFreeSystemResources</u>	USER	LIBW.LIB
<u>GetGlyphOutline</u>	GDI	LIBW.LIB
<u>GetInputState</u>	USER	LIBW.LIB
<u>GetInstanceData</u>	KERNEL	LIBW.LIB
<u>GetKBCodePage</u>	KEYBOARD	LIBW.LIB
<u>GetKerningPairs</u>	GDI	LIBW.LIB
<u>GetKeyboardState</u>	USER	LIBW.LIB
<u>GetKeyboardType</u>	KEYBOARD	LIBW.LIB
<u>GetKeyNameText</u>	KEYBOARD	LIBW.LIB
<u>GetKeyState</u>	USER	LIBW.LIB
<u>GetLastActivePopup</u>	USER	LIBW.LIB
<u>GetMapMode</u>	GDI	LIBW.LIB
<u>GetMenu</u>	USER	LIBW.LIB

<u>GetMenuCheckMarkDimensions</u>	USER	LIBW.LIB
<u>GetMenuItemCount</u>	USER	LIBW.LIB
<u>GetMenuItemID</u>	USER	LIBW.LIB
<u>GetMenuState</u>	USER	LIBW.LIB
<u>GetMenuString</u>	USER	LIBW.LIB
<u>GetMessage</u>	USER	LIBW.LIB
<u>GetMessageExtraInfo</u>	USER	LIBW.LIB
<u>GetMessagePos</u>	USER	LIBW.LIB
<u>GetMessageTime</u>	USER	LIBW.LIB
<u>GetMetaFile</u>	GDI	LIBW.LIB
<u>GetMetaFileBits</u>	GDI	LIBW.LIB
<u>GetModuleFileName</u>	KERNEL	LIBW.LIB
<u>GetModuleHandle</u>	KERNEL	LIBW.LIB
<u>GetModuleUsage</u>	KERNEL	LIBW.LIB
<u>GetNearestColor</u>	GDI	LIBW.LIB
<u>GetNearestPaletteIndex</u>	GDI	LIBW.LIB
<u>GetNextDlgGroupItem</u>	USER	LIBW.LIB
<u>GetNextDlgTabItem</u>	USER	LIBW.LIB
<u>GetNextDriver</u>	USER	LIBW.LIB
<u>GetNextWindow</u>	USER	LIBW.LIB
<u>GetNumTasks</u>	KERNEL	LIBW.LIB
<u>GetObject</u>	GDI	LIBW.LIB
<u>GetOpenClipboardWindow</u>	USER	LIBW.LIB
<u>GetOpenFileName</u>	COMMDLG	COMMDLG.LIB
<u>GetOutlineTextMetrics</u>	GDI	LIBW.LIB
<u>GetPaletteEntries</u>	GDI	LIBW.LIB
<u>GetParent</u>	USER	LIBW.LIB
<u>GetPixel</u>	GDI	LIBW.LIB
<u>GetPolyFillMode</u>	GDI	LIBW.LIB
<u>GetPriorityClipboardFormat</u>	USER	LIBW.LIB
<u>GetPrivateProfileInt</u>	KERNEL	LIBW.LIB
<u>GetPrivateProfileString</u>	KERNEL	LIBW.LIB
<u>GetProcAddress</u>	KERNEL	LIBW.LIB
<u>GetProfileInt</u>	KERNEL	LIBW.LIB
<u>GetProfileString</u>	KERNEL	LIBW.LIB
<u>GetProp</u>	USER	LIBW.LIB
<u>GetQueueStatus</u>	USER	LIBW.LIB
<u>GetRasterizerCaps</u>	GDI	LIBW.LIB
<u>GetRgnBox</u>	GDI	LIBW.LIB
<u>GetROP2</u>	GDI	LIBW.LIB
<u>GetSaveFileName</u>	COMMDLG	COMMDLG.LIB
<u>GetScrollPos</u>	USER	LIBW.LIB
<u>GetScrollRange</u>	USER	LIBW.LIB
<u>GetSelectorBase</u>	KERNEL	LIBW.LIB
<u>GetSelectorLimit</u>	KERNEL	LIBW.LIB
<u>GetStockObject</u>	GDI	LIBW.LIB
<u>GetStretchBltMode</u>	GDI	LIBW.LIB
<u>GetSubMenu</u>	USER	LIBW.LIB
<u>GetSysColor</u>	USER	LIBW.LIB
<u>GetSysModalWindow</u>	USER	LIBW.LIB
<u>GetSystemDebugState</u>	USER	LIBW.LIB
<u>GetSystemDir</u>	—	VERS.LIB



<u>GetSystemDirectory</u>	KERNEL	LIBW.LIB
<u>GetSystemMenu</u>	USER	LIBW.LIB
<u>GetSystemMetrics</u>	USER	LIBW.LIB
<u>GetSystemPaletteEntries</u>	GDI	LIBW.LIB
<u>GetSystemPaletteUse</u>	GDI	LIBW.LIB
<u>GetTabbedTextExtent</u>	USER	LIBW.LIB
<u>GetTempDrive</u>	KERNEL	LIBW.LIB
<u>GetTempFileName</u>	KERNEL	LIBW.LIB
<u>GetTextAlign</u>	GDI	LIBW.LIB
<u>GetTextCharacterExtra</u>	GDI	LIBW.LIB
<u>GetTextColor</u>	GDI	LIBW.LIB
<u>GetTextExtent</u>	GDI	LIBW.LIB
<u>GetTextExtentPoint</u>	GDI	LIBW.LIB
<u>GetTextFace</u>	GDI	LIBW.LIB
<u>GetTextMetrics</u>	GDI	LIBW.LIB
<u>GetTickCount</u>	USER	LIBW.LIB
<u>GetTimerResolution</u>	USER	LIBW.LIB
<u>GetTopWindow</u>	USER	LIBW.LIB
<u>GetUpdateRect</u>	USER	LIBW.LIB
<u>GetUpdateRgn</u>	USER	LIBW.LIB
<u>GetVersion</u>	KERNEL	LIBW.LIB
<u>GetViewportExt</u>	GDI	LIBW.LIB
<u>GetViewportExtEx</u>	GDI	LIBW.LIB
<u>GetViewportOrg</u>	GDI	LIBW.LIB
<u>GetViewportOrgEx</u>	GDI	LIBW.LIB
<u>GetWinDebugInfo</u>	KERNEL	LIBW.LIB
<u>GetWindow</u>	USER	LIBW.LIB
<u>GetWindowDC</u>	USER	LIBW.LIB
<u>GetWindowExt</u>	GDI	LIBW.LIB
<u>GetWindowExtEx</u>	GDI	LIBW.LIB
<u>GetWindowLong</u>	USER	LIBW.LIB
<u>GetWindowOrg</u>	GDI	LIBW.LIB
<u>GetWindowOrgEx</u>	GDI	LIBW.LIB
<u>GetWindowPlacement</u>	USER	LIBW.LIB
<u>GetWindowRect</u>	USER	LIBW.LIB
<u>GetWindowsDir</u>	—	VERS.LIB
<u>GetWindowsDirectory</u>	KERNEL	LIBW.LIB
<u>GetWindowTask</u>	USER	LIBW.LIB
<u>GetWindowText</u>	USER	LIBW.LIB
<u>GetWindowTextLength</u>	USER	LIBW.LIB
<u>GetWindowWord</u>	USER	LIBW.LIB
<u>GetWinFlags</u>	KERNEL	LIBW.LIB
<u>GetWinMem32Version</u>	WINMEM32	WINMEM32.LIB
<u>Global16PointerAlloc</u>	WINMEM32	WINMEM32.LIB
<u>Global16PointerFree</u>	WINMEM32	WINMEM32.LIB
<u>Global32Alloc</u>	WINMEM32	WINMEM32.LIB
<u>Global32CodeAlias</u>	WINMEM32	WINMEM32.LIB
<u>Global32CodeAliasFree</u>	WINMEM32	WINMEM32.LIB
<u>Global32Free</u>	WINMEM32	WINMEM32.LIB
<u>Global32Realloc</u>	WINMEM32	WINMEM32.LIB
<u>GlobalAddAtom</u>	USER	LIBW.LIB
<u>GlobalAlloc</u>	KERNEL	LIBW.LIB

<u>GlobalCompact</u>	KERNEL	LIBW.LIB
<u>GlobalDeleteAtom</u>	USER	LIBW.LIB
<u>GlobalDosAlloc</u>	KERNEL	LIBW.LIB
<u>GlobalDosFree</u>	KERNEL	LIBW.LIB
<u>GlobalEntryHandle</u>	TOOLHELP	TOOLHELP.LIB
<u>GlobalEntryModule</u>	TOOLHELP	TOOLHELP.LIB
<u>GlobalFindAtom</u>	USER	LIBW.LIB
<u>GlobalFirst</u>	TOOLHELP	TOOLHELP.LIB
<u>GlobalFix</u>	KERNEL	LIBW.LIB
<u>GlobalFlags</u>	KERNEL	LIBW.LIB
<u>GlobalFree</u>	KERNEL	LIBW.LIB
<u>GlobalGetAtomName</u>	USER	LIBW.LIB
<u>GlobalHandle</u>	KERNEL	LIBW.LIB
<u>GlobalHandleToSel</u>	TOOLHELP	TOOLHELP.LIB
<u>GlobalInfo</u>	TOOLHELP	TOOLHELP.LIB
<u>GlobalLock</u>	KERNEL	LIBW.LIB
<u>GlobalLRUNewest</u>	KERNEL	LIBW.LIB
<u>GlobalLRUOldest</u>	KERNEL	LIBW.LIB
<u>GlobalNext</u>	TOOLHELP	TOOLHELP.LIB
<u>GlobalNotify</u>	KERNEL	LIBW.LIB
<u>GlobalPageLock</u>	KERNEL	LIBW.LIB
<u>GlobalPageUnlock</u>	KERNEL	LIBW.LIB
<u>GlobalReAlloc</u>	KERNEL	LIBW.LIB
<u>GlobalSize</u>	KERNEL	LIBW.LIB
<u>GlobalUnfix</u>	KERNEL	LIBW.LIB
<u>GlobalUnlock</u>	KERNEL	LIBW.LIB
<u>GlobalUnWire</u>	KERNEL	LIBW.LIB
<u>GlobalWire</u>	KERNEL	LIBW.LIB
<u>GrayString</u>	USER	LIBW.LIB
<u>hardware_event</u>	USER	LIBW.LIB
<u>HelpMessageFilterHookFunction</u>	—	SCRNSAVE.LIB
<u>HideCaret</u>	USER	LIBW.LIB
<u>HiliteMenuItem</u>	USER	LIBW.LIB
<u>hmemcpy</u>	KERNEL	LIBW.LIB
<u>_hread</u>	KERNEL	LIBW.LIB
<u>_hwrite</u>	KERNEL	LIBW.LIB
<u>InflateRect</u>	USER	LIBW.LIB
<u>InitAtomTable</u>	KERNEL	LIBW.LIB
<u>InSendMessage</u>	USER	LIBW.LIB
<u>InsertMenu</u>	USER	LIBW.LIB
<u>InterruptRegister</u>	TOOLHELP	TOOLHELP.LIB
<u>InterruptUnRegister</u>	TOOLHELP	TOOLHELP.LIB
<u>IntersectClipRect</u>	GDI	LIBW.LIB
<u>IntersectRect</u>	USER	LIBW.LIB
<u>InvalidateRect</u>	USER	LIBW.LIB
<u>InvalidateRgn</u>	USER	LIBW.LIB
<u>InvertRect</u>	USER	LIBW.LIB
<u>InvertRgn</u>	GDI	LIBW.LIB
<u>IsBadCodePtr</u>	KERNEL	LIBW.LIB
<u>IsBadHugeReadPtr</u>	KERNEL	LIBW.LIB
<u>IsBadHugeWritePtr</u>	KERNEL	LIBW.LIB
<u>IsBadReadPtr</u>	KERNEL	LIBW.LIB

<u>IsBadStringPtr</u>	KERNEL	LIBW.LIB
<u>IsBadWritePtr</u>	KERNEL	LIBW.LIB
<u>IsCharAlpha</u>	USER	LIBW.LIB
<u>IsCharAlphaNumeric</u>	USER	LIBW.LIB
<u>IsCharLower</u>	USER	LIBW.LIB
<u>IsCharUpper</u>	USER	LIBW.LIB
<u>IsChild</u>	USER	LIBW.LIB
<u>IsClipboardFormatAvailable</u>	USER	LIBW.LIB
<u>IsDBCSLeadByte</u>	KERNEL	LIBW.LIB
<u>IsDialogMessage</u>	USER	LIBW.LIB
<u>IsDlgButtonChecked</u>	USER	LIBW.LIB
<u>IsGDIObject</u>	GDI	LIBW.LIB
<u>IsIconic</u>	USER	LIBW.LIB
<u>IsMenu</u>	USER	LIBW.LIB
<u>IsRectEmpty</u>	USER	LIBW.LIB
<u>IsTask</u>	KERNEL	LIBW.LIB
<u>IsWindow</u>	USER	LIBW.LIB
<u>IsWindowEnabled</u>	USER	LIBW.LIB
<u>IsWindowVisible</u>	USER	LIBW.LIB
<u>IsZoomed</u>	USER	LIBW.LIB
<u>KillTimer</u>	USER	LIBW.LIB
<u>_lclose</u>	KERNEL	LIBW.LIB
<u>_lcreat</u>	KERNEL	LIBW.LIB
<u>LimitEmsPages</u>	KERNEL	LIBW.LIB
<u>LineDDA</u>	GDI	LIBW.LIB
<u>LineTo</u>	GDI	LIBW.LIB
<u>_llseek</u>	KERNEL	LIBW.LIB
<u>LoadAccelerators</u>	USER	LIBW.LIB
<u>LoadBitmap</u>	USER	LIBW.LIB
<u>LoadCursor</u>	USER	LIBW.LIB
<u>LoadIcon</u>	USER	LIBW.LIB
<u>LoadLibrary</u>	KERNEL	LIBW.LIB
<u>LoadMenu</u>	USER	LIBW.LIB
<u>LoadMenuIndirect</u>	USER	LIBW.LIB
<u>LoadModule</u>	KERNEL	LIBW.LIB
<u>LoadResource</u>	KERNEL	LIBW.LIB
<u>LoadString</u>	USER	LIBW.LIB
<u>LocalAlloc</u>	KERNEL	LIBW.LIB
<u>LocalCompact</u>	KERNEL	LIBW.LIB
<u>LocalFirst</u>	TOOLHELP	TOOLHELP.LIB
<u>LocalFlags</u>	KERNEL	LIBW.LIB
<u>LocalFree</u>	KERNEL	LIBW.LIB
<u>LocalHandle</u>	KERNEL	LIBW.LIB
<u>LocalInfo</u>	TOOLHELP	TOOLHELP.LIB
<u>LocalInit</u>	KERNEL	LIBW.LIB
<u>LocalLock</u>	KERNEL	LIBW.LIB
<u>LocalNext</u>	TOOLHELP	TOOLHELP.LIB
<u>LocalReAlloc</u>	KERNEL	LIBW.LIB
<u>LocalShrink</u>	KERNEL	LIBW.LIB
<u>LocalSize</u>	KERNEL	LIBW.LIB
<u>LocalUnlock</u>	KERNEL	LIBW.LIB
<u>LockInput</u>	USER	LIBW.LIB

<u>LockResource</u>	KERNEL	LIBW.LIB
<u>LockSegment</u>	KERNEL	LIBW.LIB
<u>LockWindowUpdate</u>	USER	LIBW.LIB
<u>LogError</u>	KERNEL	LIBW.LIB
<u>LogParamError</u>	KERNEL	LIBW.LIB
<u>_lopen</u>	KERNEL	LIBW.LIB
<u>LPtoDP</u>	GDI	LIBW.LIB
<u>_lread</u>	KERNEL	LIBW.LIB
<u>lstrcat</u>	KERNEL	LIBW.LIB
<u>lstrcmp</u>	USER	LIBW.LIB
<u>lstrcmpi</u>	USER	LIBW.LIB
<u>lstrcpy</u>	KERNEL	LIBW.LIB
<u>lstrlen</u>	KERNEL	LIBW.LIB
<u>_lwrite</u>	KERNEL	LIBW.LIB
<u>LZClose</u>	LZEXPAND	LZEXPAND.LIB
<u>LZCopy</u>	LZEXPAND	LZEXPAND.LIB
<u>LZDone</u>	LZEXPAND	LZEXPAND.LIB
<u>LZInit</u>	LZEXPAND	LZEXPAND.LIB
<u>LZOpenFile</u>	LZEXPAND	LZEXPAND.LIB
<u>LZRead</u>	LZEXPAND	LZEXPAND.LIB
<u>LZSeek</u>	LZEXPAND	LZEXPAND.LIB
<u>LZStart</u>	LZEXPAND	LZEXPAND.LIB
<u>MakeProcInstance</u>	KERNEL	LIBW.LIB
<u>MapDialogRect</u>	USER	LIBW.LIB
<u>MapVirtualKey</u>	KEYBOARD	LIBW.LIB
<u>MapWindowPoints</u>	USER	LIBW.LIB
<u>MemManInfo</u>	TOOLHELP	TOOLHELP.LIB
<u>MemoryRead</u>	TOOLHELP	TOOLHELP.LIB
<u>MemoryWrite</u>	TOOLHELP	TOOLHELP.LIB
<u>MessageBeep</u>	USER	LIBW.LIB
<u>MessageBox</u>	USER	LIBW.LIB
<u>ModifyMenu</u>	USER	LIBW.LIB
<u>ModuleFindHandle</u>	TOOLHELP	TOOLHELP.LIB
<u>ModuleFindName</u>	TOOLHELP	TOOLHELP.LIB
<u>ModuleFirst</u>	TOOLHELP	TOOLHELP.LIB
<u>ModuleNext</u>	TOOLHELP	TOOLHELP.LIB
<u>MoveTo</u>	GDI	LIBW.LIB
<u>MoveToEx</u>	GDI	LIBW.LIB
<u>MoveWindow</u>	USER	LIBW.LIB
<u>MulDiv</u>	GDI	LIBW.LIB
<u>NetBIOSCall</u>	KERNEL	LIBW.LIB
<u>NotifyRegister</u>	TOOLHELP	TOOLHELP.LIB
<u>NotifyUnRegister</u>	TOOLHELP	TOOLHELP.LIB
<u>OemKeyScan</u>	KEYBOARD	LIBW.LIB
<u>OemToAnsi</u>	KEYBOARD	LIBW.LIB
<u>OemToAnsiBuff</u>	KEYBOARD	LIBW.LIB
<u>OffsetClipRgn</u>	GDI	LIBW.LIB
<u>OffsetRect</u>	USER	LIBW.LIB
<u>OffsetRgn</u>	GDI	LIBW.LIB
<u>OffsetViewportOrg</u>	GDI	LIBW.LIB
<u>OffsetViewportOrgEx</u>	GDI	LIBW.LIB
<u>OffsetWindowOrg</u>	GDI	LIBW.LIB

<u>OffsetWindowOrgEx</u>	GDI	LIBW.LIB
<u>OleActivate</u>	OLECLI	OLECLI.LIB
<u>OleBlockServer</u>	OLESVR	OLESVR.LIB
<u>OleClone</u>	OLECLI	OLECLI.LIB
<u>OleClose</u>	OLECLI	OLECLI.LIB
<u>OleCopyFromLink</u>	OLECLI	OLECLI.LIB
<u>OleCopyToClipboard</u>	OLECLI	OLECLI.LIB
<u>OleCreate</u>	OLECLI	OLECLI.LIB
<u>OleCreateFromClip</u>	OLECLI	OLECLI.LIB
<u>OleCreateFromFile</u>	OLECLI	OLECLI.LIB
<u>OleCreateFromTemplate</u>	OLECLI	OLECLI.LIB
<u>OleCreateInvisible</u>	OLECLI	OLECLI.LIB
<u>OleCreateLinkFromClip</u>	OLECLI	OLECLI.LIB
<u>OleCreateLinkFromFile</u>	OLECLI	OLECLI.LIB
<u>OleDelete</u>	OLECLI	OLECLI.LIB
<u>OleDraw</u>	OLECLI	OLECLI.LIB
<u>OleEnumFormats</u>	OLECLI	OLECLI.LIB
<u>OleEnumObjects</u>	OLECLI	OLECLI.LIB
<u>OleEqual</u>	OLECLI	OLECLI.LIB
<u>OleExecute</u>	OLECLI	OLECLI.LIB
<u>OleGetData</u>	OLECLI	OLECLI.LIB
<u>OleGetLinkUpdateOptions</u>	OLECLI	OLECLI.LIB
<u>OleIsDcMeta</u>	OLECLI	OLECLI.LIB
<u>OleLoadFromStream</u>	OLECLI	OLECLI.LIB
<u>OleLockServer</u>	OLECLI	OLECLI.LIB
<u>OleObjectConvert</u>	OLECLI	OLECLI.LIB
<u>OleQueryBounds</u>	OLECLI	OLECLI.LIB
<u>OleQueryClientVersion</u>	OLECLI	OLECLI.LIB
<u>OleQueryCreateFromClip</u>	OLECLI	OLECLI.LIB
<u>OleQueryLinkFromClip</u>	OLECLI	OLECLI.LIB
<u>OleQueryName</u>	OLECLI	OLECLI.LIB
<u>OleQueryOpen</u>	OLECLI	OLECLI.LIB
<u>OleQueryOutOfDate</u>	OLECLI	OLECLI.LIB
<u>OleQueryProtocol</u>	OLECLI	OLECLI.LIB
<u>OleQueryReleaseError</u>	OLECLI	OLECLI.LIB
<u>OleQueryReleaseMethod</u>	OLECLI	OLECLI.LIB
<u>OleQueryReleaseStatus</u>	OLECLI	OLECLI.LIB
<u>OleQueryServerVersion</u>	OLESVR	OLESVR.LIB
<u>OleQuerySize</u>	OLECLI	OLECLI.LIB
<u>OleQueryType</u>	OLECLI	OLECLI.LIB
<u>OleReconnect</u>	OLECLI	OLECLI.LIB
<u>OleRegisterClientDoc</u>	OLECLI	OLECLI.LIB
<u>OleRegisterServer</u>	OLESVR	OLESVR.LIB
<u>OleRegisterServerDoc</u>	OLESVR	OLESVR.LIB
<u>OleRelease</u>	OLECLI	OLECLI.LIB
<u>OleRename</u>	OLECLI	OLECLI.LIB
<u>OleRenameClientDoc</u>	OLECLI	OLECLI.LIB
<u>OleRenameServerDoc</u>	OLESVR	OLESVR.LIB
<u>OleRequestData</u>	OLECLI	OLECLI.LIB
<u>OleRevertClientDoc</u>	OLECLI	OLECLI.LIB
<u>OleRevertServerDoc</u>	OLESVR	OLESVR.LIB
<u>OleRevokeClientDoc</u>	OLECLI	OLECLI.LIB

<u>OleRevokeObject</u>	OLESVR	OLESVR.LIB
<u>OleRevokeServer</u>	OLESVR	OLESVR.LIB
<u>OleRevokeServerDoc</u>	OLESVR	OLESVR.LIB
<u>OleSavedClientDoc</u>	OLECLI	OLECLI.LIB
<u>OleSavedServerDoc</u>	OLESVR	OLESVR.LIB
<u>OleSaveToStream</u>	OLECLI	OLECLI.LIB
<u>OleSetBounds</u>	OLECLI	OLECLI.LIB
<u>OleSetColorScheme</u>	OLECLI	OLECLI.LIB
<u>OleSetData</u>	OLECLI	OLECLI.LIB
<u>OleSetHostNames</u>	OLECLI	OLECLI.LIB
<u>OleSetLinkUpdateOptions</u>	OLECLI	OLECLI.LIB
<u>OleSetTargetDevice</u>	OLECLI	OLECLI.LIB
<u>OleUnblockServer</u>	OLESVR	OLESVR.LIB
<u>OleUnlockServer</u>	OLECLI	OLECLI.LIB
<u>OleUpdate</u>	OLECLI	OLECLI.LIB
<u>OpenClipboard</u>	USER	LIBW.LIB
<u>OpenComm</u>	USER	LIBW.LIB
<u>OpenDriver</u>	USER	LIBW.LIB
<u>OpenFile</u>	LZEXPAND	LZEXPAND.LIB
<u>OpenIcon</u>	USER	LIBW.LIB
<u>OutputDebugString</u>	KERNEL	LIBW.LIB
<u>PaintRgn</u>	GDI	LIBW.LIB
<u>PatBlt</u>	GDI	LIBW.LIB
<u>PeekMessage</u>	USER	LIBW.LIB
<u>Pie</u>	GDI	LIBW.LIB
<u>PlayMetaFile</u>	GDI	LIBW.LIB
<u>PlayMetaFileRecord</u>	GDI	LIBW.LIB
<u>Polygon</u>	GDI	LIBW.LIB
<u>Polyline</u>	GDI	LIBW.LIB
<u>PolyPolygon</u>	GDI	LIBW.LIB
<u>PostAppMessage</u>	USER	LIBW.LIB
<u>PostMessage</u>	USER	LIBW.LIB
<u>PostQuitMessage</u>	USER	LIBW.LIB
<u>PrestoChangoSelector</u>	KERNEL	LIBW.LIB
<u>PrintDlg</u>	COMMMDLG	COMMMDLG.LIB
<u>ProfClear</u>	—	LIBW.LIB
<u>ProfFinish</u>	—	LIBW.LIB
<u>ProfFlush</u>	—	LIBW.LIB
<u>ProfInsChk</u>	—	LIBW.LIB
<u>ProfSampRate</u>	—	LIBW.LIB
<u>ProfSetup</u>	—	LIBW.LIB
<u>ProfStart</u>	—	LIBW.LIB
<u>ProfStop</u>	—	LIBW.LIB
<u>PtInRect</u>	USER	LIBW.LIB
<u>PtInRegion</u>	GDI	LIBW.LIB
<u>PtVisible</u>	GDI	LIBW.LIB
<u>QueryAbort</u>	GDI	LIBW.LIB
<u>QuerySendMessage</u>	USER	LIBW.LIB
<u>ReadComm</u>	USER	LIBW.LIB
<u>RealizePalette</u>	USER	LIBW.LIB
<u>Rectangle</u>	GDI	LIBW.LIB
<u>RectInRegion</u>	GDI	LIBW.LIB

<u>RectVisible</u>	GDI	LIBW.LIB
<u>RedrawWindow</u>	USER	LIBW.LIB
<u>RegCloseKey</u>	SHELL	SHELL.LIB
<u>RegCreateKey</u>	SHELL	SHELL.LIB
<u>RegDeleteKey</u>	SHELL	SHELL.LIB
<u>RegEnumKey</u>	SHELL	SHELL.LIB
<u>RegisterClass</u>	USER	LIBW.LIB
<u>RegisterClipboardFormat</u>	USER	LIBW.LIB
<u>RegisterWindowMessage</u>	USER	LIBW.LIB
<u>RegOpenKey</u>	SHELL	SHELL.LIB
<u>RegQueryValue</u>	SHELL	SHELL.LIB
<u>RegSetValue</u>	SHELL	SHELL.LIB
<u>ReleaseCapture</u>	USER	LIBW.LIB
<u>ReleaseDC</u>	USER	LIBW.LIB
<u>RemoveFontResource</u>	GDI	LIBW.LIB
<u>RemoveMenu</u>	USER	LIBW.LIB
<u>RemoveProp</u>	USER	LIBW.LIB
<u>ReplaceText</u>	COMMDDL	COMMDDL.LIB
<u>ReplyMessage</u>	USER	LIBW.LIB
<u>ResetDC</u>	GDI	LIBW.LIB
<u>ResizePalette</u>	GDI	LIBW.LIB
<u>RestoreDC</u>	GDI	LIBW.LIB
<u>RoundRect</u>	GDI	LIBW.LIB
<u>SaveDC</u>	GDI	LIBW.LIB
<u>ScaleViewportExt</u>	GDI	LIBW.LIB
<u>ScaleViewportExtEx</u>	GDI	LIBW.LIB
<u>ScaleWindowExt</u>	GDI	LIBW.LIB
<u>ScaleWindowExtEx</u>	GDI	LIBW.LIB
<u>ScreenSaverProc</u>	—	SCRNSAVE.LIB
<u>ScreenToClient</u>	USER	LIBW.LIB
<u>ScrollDC</u>	USER	LIBW.LIB
<u>ScrollWindow</u>	USER	LIBW.LIB
<u>ScrollWindowEx</u>	USER	LIBW.LIB
<u>SelectClipRgn</u>	GDI	LIBW.LIB
<u>SelectObject</u>	GDI	LIBW.LIB
<u>SelectPalette</u>	USER	LIBW.LIB
<u>SendDlgItemMessage</u>	USER	LIBW.LIB
<u>SendDriverMessage</u>	USER	LIBW.LIB
<u>SendMessage</u>	USER	LIBW.LIB
<u>SetAbortProc</u>	GDI	LIBW.LIB
<u>SetActiveWindow</u>	USER	LIBW.LIB
<u>SetBitmapBits</u>	GDI	LIBW.LIB
<u>SetBitmapDimension</u>	GDI	LIBW.LIB
<u>SetBitmapDimensionEx</u>	GDI	LIBW.LIB
<u>SetBkColor</u>	GDI	LIBW.LIB
<u>SetBkMode</u>	GDI	LIBW.LIB
<u>SetBoundsRect</u>	GDI	LIBW.LIB
<u>SetBrushOrg</u>	GDI	LIBW.LIB
<u>SetCapture</u>	USER	LIBW.LIB
<u>SetCaretBlinkTime</u>	USER	LIBW.LIB
<u>SetCaretPos</u>	USER	LIBW.LIB
<u>SetClassLong</u>	USER	LIBW.LIB

<u>SetClassWord</u>	USER	LIBW.LIB
<u>SetClipboardData</u>	USER	LIBW.LIB
<u>SetClipboardViewer</u>	USER	LIBW.LIB
<u>SetCommBreak</u>	USER	LIBW.LIB
<u>SetCommEventMask</u>	USER	LIBW.LIB
<u>SetCommState</u>	USER	LIBW.LIB
<u>SetCursor</u>	USER	LIBW.LIB
<u>SetCursorPos</u>	USER	LIBW.LIB
<u>SetDIBits</u>	GDI	LIBW.LIB
<u>SetDIBitsToDevice</u>	GDI	LIBW.LIB
<u>SetDlgItemInt</u>	USER	LIBW.LIB
<u>SetDlgItemText</u>	USER	LIBW.LIB
<u>SetDoubleClickTime</u>	USER	LIBW.LIB
<u>SetErrorMode</u>	KERNEL	LIBW.LIB
<u>SetFocus</u>	USER	LIBW.LIB
<u>SetHandleCount</u>	KERNEL	LIBW.LIB
<u>SetKeyboardState</u>	USER	LIBW.LIB
<u>SetMapMode</u>	GDI	LIBW.LIB
<u>SetMapperFlags</u>	GDI	LIBW.LIB
<u>SetMenu</u>	USER	LIBW.LIB
<u>SetMenuItemBitmaps</u>	USER	LIBW.LIB
<u>SetMessageQueue</u>	USER	LIBW.LIB
<u>SetMetaFileBits</u>	GDI	LIBW.LIB
<u>SetMetaFileBitsBetter</u>	GDI	LIBW.LIB
<u>SetPaletteEntries</u>	GDI	LIBW.LIB
<u>SetParent</u>	USER	LIBW.LIB
<u>SetPixel</u>	GDI	LIBW.LIB
<u>SetPolyFillMode</u>	GDI	LIBW.LIB
<u>SetProp</u>	USER	LIBW.LIB
<u>SetRect</u>	USER	LIBW.LIB
<u>SetRectEmpty</u>	USER	LIBW.LIB
<u>SetRectRgn</u>	GDI	LIBW.LIB
<u>SetResourceHandler</u>	KERNEL	LIBW.LIB
<u>SetROP2</u>	GDI	LIBW.LIB
<u>SetScrollPos</u>	USER	LIBW.LIB
<u>SetScrollRange</u>	USER	LIBW.LIB
<u>SetSelectorBase</u>	KERNEL	LIBW.LIB
<u>SetSelectorLimit</u>	KERNEL	LIBW.LIB
<u>SetStretchBltMode</u>	GDI	LIBW.LIB
<u>SetSwapAreaSize</u>	KERNEL	LIBW.LIB
<u>SetSysColors</u>	USER	LIBW.LIB
<u>SetSysModalWindow</u>	USER	LIBW.LIB
<u>SetSystemPaletteUse</u>	GDI	LIBW.LIB
<u>SetTextAlign</u>	GDI	LIBW.LIB
<u>SetTextCharacterExtra</u>	GDI	LIBW.LIB
<u>SetTextColor</u>	GDI	LIBW.LIB
<u>SetTextJustification</u>	GDI	LIBW.LIB
<u>SetTimer</u>	USER	LIBW.LIB
<u>SetViewportExt</u>	GDI	LIBW.LIB
<u>SetViewportExtEx</u>	GDI	LIBW.LIB
<u>SetViewportOrg</u>	GDI	LIBW.LIB
<u>SetViewportOrgEx</u>	GDI	LIBW.LIB



<u>SetWinDebugInfo</u>	KERNEL	LIBW.LIB
<u>SetWindowExt</u>	GDI	LIBW.LIB
<u>SetWindowExtEx</u>	GDI	LIBW.LIB
<u>SetWindowLong</u>	USER	LIBW.LIB
<u>SetWindowOrg</u>	GDI	LIBW.LIB
<u>SetWindowOrgEx</u>	GDI	LIBW.LIB
<u>SetWindowPlacement</u>	USER	LIBW.LIB
<u>SetWindowPos</u>	USER	LIBW.LIB
<u>SetWindowsHook</u>	USER	LIBW.LIB
<u>SetWindowsHookEx</u>	USER	LIBW.LIB
<u>SetWindowText</u>	USER	LIBW.LIB
<u>SetWindowWord</u>	USER	LIBW.LIB
<u>ShellExecute</u>	SHELL	SHELL.LIB
<u>ShowCaret</u>	USER	LIBW.LIB
<u>ShowCursor</u>	USER	LIBW.LIB
<u>ShowOwnedPopups</u>	USER	LIBW.LIB
<u>ShowScrollBar</u>	USER	LIBW.LIB
<u>ShowWindow</u>	USER	LIBW.LIB
<u>SizeofResource</u>	KERNEL	LIBW.LIB
<u>SpoolFile</u>	GDI	LIBW.LIB
<u>StackTraceCSIPFirst</u>	TOOLHELP	TOOLHELP.LIB
<u>StackTraceFirst</u>	TOOLHELP	TOOLHELP.LIB
<u>StackTraceNext</u>	TOOLHELP	TOOLHELP.LIB
<u>StartDoc</u>	GDI	LIBW.LIB
<u>StartPage</u>	GDI	LIBW.LIB
<u>StretchBlt</u>	GDI	LIBW.LIB
<u>StretchDIBits</u>	GDI	LIBW.LIB
<u>SubtractRect</u>	USER	LIBW.LIB
<u>SwapMouseButton</u>	USER	LIBW.LIB
<u>SwapRecording</u>	KERNEL	LIBW.LIB
<u>SwitchStackBack</u>	KERNEL	LIBW.LIB
<u>SwitchStackTo</u>	KERNEL	LIBW.LIB
<u>SystemHeapInfo</u>	TOOLHELP	TOOLHELP.LIB
<u>SystemParametersInfo</u>	USER	LIBW.LIB
<u>TabbedTextOut</u>	USER	LIBW.LIB
<u>TaskFindHandle</u>	TOOLHELP	TOOLHELP.LIB
<u>TaskFirst</u>	TOOLHELP	TOOLHELP.LIB
<u>TaskGetCSIP</u>	TOOLHELP	TOOLHELP.LIB
<u>TaskNext</u>	TOOLHELP	TOOLHELP.LIB
<u>TaskSetCSIP</u>	TOOLHELP	TOOLHELP.LIB
<u>TaskSwitch</u>	TOOLHELP	TOOLHELP.LIB
<u>TerminateApp</u>	TOOLHELP	TOOLHELP.LIB
<u>TextOut</u>	GDI	LIBW.LIB
<u>Throw</u>	KERNEL	LIBW.LIB
<u>TimerCount</u>	TOOLHELP	TOOLHELP.LIB
<u>ToAscii</u>	KEYBOARD	LIBW.LIB
<u>TrackPopupMenu</u>	USER	LIBW.LIB
<u>TranslateAccelerator</u>	USER	LIBW.LIB
<u>TranslateMDISysAccel</u>	USER	LIBW.LIB
<u>TranslateMessage</u>	USER	LIBW.LIB
<u>TransmitCommChar</u>	USER	LIBW.LIB
<u>UnAllocDiskSpace</u>	STRESS	STRESS.LIB

<u>UnAllocFileHandles</u>	STRESS	STRESS.LIB
<u>UngetCommChar</u>	USER	LIBW.LIB
<u>UnhookWindowsHook</u>	USER	LIBW.LIB
<u>UnhookWindowsHookEx</u>	USER	LIBW.LIB
<u>UnionRect</u>	USER	LIBW.LIB
<u>UnlockSegment</u>	KERNEL	LIBW.LIB
<u>UnrealizeObject</u>	GDI	LIBW.LIB
<u>UnregisterClass</u>	USER	LIBW.LIB
<u>UpdateColors</u>	GDI	LIBW.LIB
<u>UpdateWindow</u>	USER	LIBW.LIB
<u>ValidateCodeSegments</u>	KERNEL	LIBW.LIB
<u>ValidateFreeSpaces</u>	KERNEL	LIBW.LIB
<u>ValidateRect</u>	USER	LIBW.LIB
<u>ValidateRgn</u>	USER	LIBW.LIB
<u>VerFindFile</u>	VER	VER.LIB
<u>VerInstallFile</u>	VER	VER.LIB
<u>VerLanguageName</u>	VER	VER.LIB
<u>VerQueryValue</u>	VER	VER.LIB
<u>VkKeyScan</u>	KEYBOARD	LIBW.LIB
<u>WaitMessage</u>	USER	LIBW.LIB
<u>WindowFromPoint</u>	USER	LIBW.LIB
<u>WinExec</u>	KERNEL	LIBW.LIB
<u>WinHelp</u>	USER	LIBW.LIB
<u>WNetAddConnection</u>	USER	LIBW.LIB
<u>WNetCancelConnection</u>	USER	LIBW.LIB
<u>WNetGetConnection</u>	USER	LIBW.LIB
<u>WriteComm</u>	USER	LIBW.LIB
<u>WritePrivateProfileString</u>	KERNEL	LIBW.LIB
<u>WriteProfileString</u>	KERNEL	LIBW.LIB
<u>_wsprintf</u>	USER	LIBW.LIB
<u>wvsprintf</u>	USER	LIBW.LIB
<u>Yield</u>	KERNEL	LIBW.LIB

## Binary and Ternary Raster-Operation Codes

This topic lists and describes the binary and ternary raster operations used by graphics device interface (GDI). A binary raster operation involves two operands: a pen and a destination bitmap. A ternary raster operation involves three operands: a source bitmap, a brush, and a destination bitmap. Both binary and ternary raster operations use Boolean operators.

### Binary Raster Operations

This section lists the binary raster-operation codes used by the `GetROP2` and `SetROP2` functions. Raster-operation codes define how GDI combines the bits from the selected pen with the bits in the destination bitmap.

Each raster-operation code represents a Boolean operation in which the values of the pixels in the selected pen and the destination bitmap are combined. Following are the two operands used in these operations:

Operand	Meaning
P	Selected pen
D	Destination bitmap

The Boolean operators used in these operations follow:

Operator	Meaning
a	Bitwise AND
n	Bitwise NOT (inverse)
o	Bitwise OR
x	Bitwise exclusive OR (XOR)

All Boolean operations are presented in reverse Polish notation. For example, the following operation replaces the values of the pixels in the destination bitmap with a combination of the pixel values of the pen and the selected brush:

`DPo`

Each raster-operation code is a 32-bit integer whose high-order word is a Boolean operation index and whose low-order word is the operation code. The 16-bit operation index is a zero-extended 8-bit value that represents all possible outcomes resulting from the Boolean operation on two parameters (in this case, the pen and destination values). For example, the operation indexes for the `DPo` and `DPan` operations are shown in the following list:

P	D	DPo	DPan
0	0	0	1
0	1	1	1
1	0	1	1
1	1	1	0

The following list outlines the drawing modes and the Boolean operations that they represent:

Raster operation	Boolean operation
<code>R2_BLACK</code>	0
<code>R2_COPYPEN</code>	P
<code>R2_MASKNOTPEN</code>	DPna
<code>R2_MASKPEN</code>	DPa
<code>R2_MASKPENNOT</code>	PDna
<code>R2_MERGEOTPEN</code>	DPno
<code>R2_MERGEPEPEN</code>	DPo
<code>R2_MERGEPEPENNOT</code>	PDno
<code>R2_NOP</code>	D
<code>R2_NOT</code>	Dn
<code>R2_NOTCOPYPEN</code>	Pn
<code>R2_NOTMASKPEN</code>	DPan
<code>R2_NOTMERGEPEPEN</code>	DPon
<code>R2_NOTXORPEN</code>	DPxn

R2_WHITE	1
R2_XORPEN	DPx

For a monochrome device, GDI maps the value zero to black and the value 1 to white. If an application attempts to draw with a black pen on a white destination by using the available binary raster operations, the following results occur:

Raster operation	Result
R2_BLACK	Visible black line
R2_COPYPEN	Visible black line
R2_MASKNOTPEN	No visible line
R2_MASKPEN	Visible black line
R2_MASKPENNOT	Visible black line
R2_MERGENOTPEN	No visible line
R2_MERGEPEPEN	Visible black line
R2_MERGEPEPENNOT	Visible black line
R2_NOP	No visible line
R2_NOT	Visible black line
R2_NOTCOPYPEN	No visible line
R2_NOTMASKPEN	No visible line
R2_NOTMERGEPEPEN	Visible black line
R2_NOTXORPEN	Visible black line
R2_WHITE	No visible line
R2_XORPEN	No visible line

For a color device, GDI uses RGB values to represent the colors of the pen and the destination. An RGB color value is a long integer that contains a red, a green, and a blue color field, each specifying the intensity of the given color. Intensities range from 0 through 255. The values are packed in the three low-order bytes of the long integer. The color of a pen is always a solid color, but the color of the destination may be a mixture of any two or three colors. If an application attempts to draw with a white pen on a blue destination by using the available binary raster operations, the following results occur:

Raster operation	Result
R2_BLACK	Visible black line
R2_COPYPEN	Visible white line
R2_MASKNOTPEN	Visible black line
R2_MASKPEN	Invisible blue line
R2_MASKPENNOT	Visible red/green line
R2_MERGENOTPEN	Invisible blue line
R2_MERGEPEPEN	Visible white line
R2_MERGEPEPENNOT	Visible white line
R2_NOP	Invisible blue line
R2_NOT	Visible red/green line
R2_NOTCOPYPEN	Visible black line
R2_NOTMASKPEN	Visible red/green line
R2_NOTMERGEPEPEN	Visible black line
R2_NOTXORPEN	Invisible blue line
R2_WHITE	Visible white line
R2_XORPEN	Visible red/green line

### Ternary Raster Operations

This section lists the ternary raster-operation codes used by the BitBlt, PatBlt, and StretchBlt functions. Ternary raster-operation codes define how GDI combines the bits in a source bitmap with the bits in the destination bitmap.

Each raster-operation code represents a Boolean operation in which the values of the pixels in the source, the selected brush, and the destination are combined. Following are the three operands used in these operations:

Operand	Meaning
---------	---------

D Destination bitmap  
 P Selected brush (also called pattern)  
 S Source bitmap

Boolean operators used in these operations follow:

Operator	Meaning
a	Bitwise AND
n	Bitwise NOT (inverse)
o	Bitwise OR
x	Bitwise exclusive OR (XOR)

All Boolean operations are presented in reverse Polish notation. For example, the following operation replaces the values of the pixels in the destination bitmap with a combination of the pixel values of the source and brush:

PSo

The following operation combines the values of the pixels in the source and brush with the pixel values of the destination bitmap (there are alternative spellings of the same function, so although a particular spelling may not be in the list, an equivalent form would be):

DPSoo

Each raster-operation code is a 32-bit integer whose high-order word is a Boolean operation index and whose low-order word is the operation code. The 16-bit operation index is a zero-extended, 8-bit value that represents the result of the Boolean operation on predefined brush, source, and destination values. For example, the operation indexes for the PSo and DPSoo operations are shown in the following list:

P	S	D	PSo	DPSoo
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Operation index:   00FCh   00FEh

In this case, PSo has the operation index 00FC (read from the bottom up); DPSoo has the operation index 00FE. These values define the location of the corresponding raster-operation codes, as shown in Table A. 1, "Raster-Operation Codes." The PSo operation is in line 252 (00FCh) of the table; DPSoo is in line 254 (00FEh).

The most commonly used raster operations have been given special names in the Windows include file, WINDOWS.H. You should use these names whenever possible in your applications.

When the source and destination bitmaps are monochrome, a bit value of zero represents a black pixel and a bit value of 1 represents a white pixel. When the source and the destination bitmaps are color, those colors are represented with RGB values. For more information about RGB values, see the RGB structure in Chapter 3, "Structures."

Boolean function (hexadecimal)	Raster operation (hexadecimal)	Boolean function in reverse Polish	Common name
00	00000042	0	BLACKNESS
01	00010289	DPSoon	-
02	00020C89	DPSona	-
03	000300AA	PSon	-
04	00040C88	SDPona	-
05	000500A9	DPon	-
06	00060865	PDSxnon	-
07	000702C5	PDSaon	-
08	00080F08	SDPnaa	-

09	00090245	PDSxon	-
0A	000A0329	DPna	-
0B	000B0B2A	PSDnaon	-
0C	000C0324	SPna	-
0D	000D0B25	PDSnaon	-
0E	000E08A5	PDSonon	-
0F	000F0001	Pn	-
10	00100C85	PDSona	-
11	001100A6	DSon	NOTSRCERASE
12	00120868	SDPxnon	-
13	001302C8	SDPaon	-
14	00140869	DPSxon	-
15	001502C9	DPSaon	-
16	00165CCA	PSDPSanaxx	-
17	00171D54	SSPxDSxaxn	-
18	00180D59	SPxPDxa	-
19	00191CC8	SDPSanaxn	-
1A	001A06C5	PDSPaox	-
1B	001B0768	SDPSxaxn	-
1C	001C06CA	PSDPaox	-
1D	001D0766	DSPDXaxn	-
1E	001E01A5	PDSox	-
1F	001F0385	PDSoan	-
20	00200F09	DPSnaa	-
21	00210248	SDPxon	-
22	00220326	DSna	-
23	00230B24	SPDnaon	-
24	00240D55	SPxDSxa	-
25	00251CC5	PDSPanaxn	-
26	002606C8	SDPSaox	-
27	00271868	SDPSxnox	-
28	00280369	DPSxa	-
29	002916CA	PSDPSaoxxn	-
2A	002A0CC9	DPSana	-
2B	002B1D58	SSPxPDxaxn	-
2C	002C0784	SPDSoax	-
2D	002D060A	PSDnox	-
2E	002E064A	PSDPxox	-
2F	002F0E2A	PSDnoan	-
30	0030032A	PSna	-
31	00310B28	SDPnaon	-
32	00320688	SDPSoox	-
33	00330008	Sn	NOTSRCCOPY
34	003406C4	SPDSaox	-
35	00351864	SPDSxnox	-
36	003601A8	SDPox	-
37	00370388	SDPoan	-
38	0038078A	PSDPoax	-
39	00390604	SPDnox	-
3A	003A0644	SPDSxox	-
3B	003B0E24	SPDnoan	-
3C	003C004A	PSx	-

3D	003D18A4	SPDSonox	-
3E	003E1B24	SPDSnaox	-
3F	003F00EA	PSan	-
40	00400F0A	PSDnaa	-
41	00410249	DPSxon	-
42	00420D5D	SDxPDxa	-
43	00431CC4	SPDSanaxn	-
44	00440328	SDna	SRCERASE
45	00450B29	DPSnaon	-
46	004606C6	DSPDaox	-
47	0047076A	PSDPxaxn	-
48	00480368	SDPxa	-
49	004916C5	PDSPDaoxxn	-
4A	004A0789	DPSDoax	-
4B	004B0605	PDSnox	-
4C	004C0CC8	SDPana	-
4D	004D1954	SSPxDSxoxn	-
4E	004E0645	PDSPxox	-
4F	004F0E25	PDSnoan	-
50	00500325	PDna	-
51	00510B26	DSPnaon	-
52	005206C9	DPSDaox	-
53	00530764	SPDSxaxn	-
54	005408A9	DPSonon	-
55	00550009	Dn	DSTINVERT
56	005601A9	DPSox	-
57	00570389	DPSoan	-
58	00580785	PDSPoax	-
59	00590609	DPSnox	-
5A	005A0049	DPx	PATINVERT
5B	005B18A9	DPSDonox	-
5C	005C0649	DPSDxox	-
5D	005D0E29	DPSnoan	-
5E	005E1B29	DPSDnaox	-
5F	005F00E9	DPan	-
60	00600365	PDSxa	-
61	006116C6	DSPDSaoxxn	-
62	00620786	DSPDoax	-
63	00630608	SDPnox	-
64	00640788	SDPSoax	-
65	00650606	DSPnox	-
66	00660046	DSx	SRCINVERT
67	006718A8	SDPSonox	-
68	006858A6	DSPDSonoxxn	-
69	00690145	PDSxxn	-
6A	006A01E9	DPSax	-
6B	006B178A	PSDPSoaxxn	-
6C	006C01E8	SDPax	-
6D	006D1785	PDSPDoaxxn	-
6E	006E1E28	SDPSnoax	-
6F	006F0C65	PDSxnan	-
70	00700CC5	PDSana	-

71	00711D5C	SSDxPDxaxn	-
72	00720648	SDPSxox	-
73	00730E28	SDPnoan	-
74	00740646	DSPDxox	-
75	00750E26	DSPnoan	-
76	00761B28	SDPSnaox	-
77	007700E6	DSan	-
78	007801E5	PDSax	-
79	00791786	DSPDSoaxxn	-
7A	007A1E29	DPSDnoax	-
7B	007B0C68	SDPxnan	-
7C	007C1E24	SPDSnoax	-
7D	007D0C69	DPSxnan	-
7E	007E0955	SPxDSxo	-
7F	007F03C9	DPSaan	-
80	008003E9	DPSaa	-
81	00810975	SPxDSxon	-
82	00820C49	DPSxna	-
83	00831E04	SPDSnoaxn	-
84	00840C48	SDPxna	-
85	00851E05	PDSPnoaxn	-
86	008617A6	DSPDSoaxx	-
87	008701C5	PDSaxn	-
88	008800C6	DSa	SRCAND
89	00891B08	SDPSnaoxn	-
8A	008A0E06	DSPnoa	-
8B	008B0666	DSPDxoxn	-
8C	008C0E08	SDPnoa	-
8D	008D0668	SDPSxoxn	-
8E	008E1D7C	SSDxPDxax	-
8F	008F0CE5	PDSanan	-
90	00900C45	PDSxna	-
91	00911E08	SDPSnoaxn	-
92	009217A9	DPSDPoaxx	-
93	009301C4	SPDaxn	-
94	009417AA	PSDPSoaxx	-
95	009501C9	DPSaxn	-
96	00960169	DPSxx	-
97	0097588A	PSDPSonoxx	-
98	00981888	SDPSonoxn	-
99	00990066	DSxn	-
9A	009A0709	DPSnax	-
9B	009B07A8	SDPSoaxn	-
9C	009C0704	SPDnax	-
9D	009D07A6	DSPDoaxn	-
9E	009E16E6	DSPDSoaxx	-
9F	009F0345	PDSxan	-
A0	00A000C9	DPa	-
A1	00A11B05	PDSPnaoxn	-
A2	00A20E09	DPSnoa	-
A3	00A30669	DPSDxoxn	-
A4	00A41885	PDSPonoxn	-



A5	00A50065	PDxn	-
A6	00A60706	DSPnax	-
A7	00A707A5	PDSPoaxn	-
A8	00A803A9	DPSoa	-
A9	00A90189	DPSoxn	-
AA	00AA0029	D	-
AB	00AB0889	DPSono	-
AC	00AC0744	SPDSxax	-
AD	00AD06E9	DPSDaoxn	-
AE	00AE0B06	DSPnao	-
AF	00AF0229	DPno	-
B0	00B00E05	PDSnoa	-
B1	00B10665	PDSPxoxn	-
B2	00B21974	SSPxDSxox	-
B3	00B30CE8	SDPanan	-
B4	00B4070A	PSDnax	-
B5	00B507A9	DPSDoaxn	-
B6	00B616E9	DPSDPaoxx	-
B7	00B70348	SDPxan	-
B8	00B8074A	PSDPxax	-
B9	00B906E6	DSPDaoxn	-
BA	00BA0B09	DPSnao	-
BB	00BB0226	DSno	MERGEPAINT
BC	00BC1CE4	SPDSanax	-
BD	00BD0D7D	SDxPDxan	-
BE	00BE0269	DPSxo	-
BF	00BF08C9	DPSano	-
C0	00C000CA	PSa	MERGECOPY
C1	00C11B04	SPDSnaoxn	-
C2	00C21884	SPDSonoxn	-
C3	00C3006A	PSxn	-
C4	00C40E04	SPDnoa	-
C5	00C50664	SPDSxoxn	-
C6	00C60708	SDPnax	-
C7	00C707AA	PSDPoaxn	-
C8	00C803A8	SDPoa	-
C9	00C90184	SPDoxn	-
CA	00CA0749	DPSDxax	-
CB	00CB06E4	SPDSaoxn	-
CC	00CC0020	S	SRCCOPY
CD	00CD0888	SDPono	-
CE	00CE0B08	SDPnao	-
CF	00CF0224	SPno	-
D0	00D00E0A	PSDnoa	-
D1	00D1066A	PSDPxoxn	-
D2	00D20705	PDSnax	-
D3	00D307A4	SPDSoaxn	-
D4	00D41D78	SSPxPDxax	-
D5	00D50CE9	DPSanan	-
D6	00D616EA	PSDPSaoxx	-
D7	00D70349	DPSxan	-
D8	00D80745	PDSPxax	-

D9	00D906E8	SDPSaoxn	-
DA	00DA1CE9	DPSDanax	-
DB	00DB0D75	SPxDSxan	-
DC	00DC0B04	SPDnao	-
DD	00DD0228	SDno	-
DE	00DE0268	SDPxO	-
DF	00DF08C8	SDPano	-
E0	00E003A5	PDSoa	-
E1	00E10185	PDSoxn	-
E2	00E20746	DSPDxax	-
E3	00E306EA	PSDPaoxn	-
E4	00E40748	SDPSxax	-
E5	00E506E5	PDSPaoxn	-
E6	00E61CE8	SDPSanax	-
E7	00E70D79	SPxPDxan	-
E8	00E81D74	SSPxDSxax	-
E9	00E95CE6	DSPDSanaxxn	-
EA	00EA02E9	DPSao	-
EB	00EB0849	DPSxno	-
EC	00EC02E8	SDPao	-
ED	00ED0848	SDPxno	-
EE	00EE0086	DSO	SRCPAINT
EF	00EF0A08	SDPnoo	-
F0	00F00021	P	PATCOPY
F1	00F10885	PDSono	-
F2	00F20B05	PDSnao	-
F3	00F3022A	PSno	-
F4	00F40B0A	PSDnao	-
F5	00F50225	PDno	-
F6	00F60265	PDSxo	-
F7	00F708C5	PDSano	-
F8	00F802E5	PDSao	-
F9	00F90845	PDSxno	-
FA	00FA0089	DPo	-
FB	00FB0A09	DPSnoo	PATPAINT
FC	00FC008A	PSO	-
FD	00FD0A0A	PSDnoo	-
FE	00FE02A9	DPSoo	-
FF	00FF0062	1	WHITENESS

## Resource Compiler Diagnostic Messages

This topic contains descriptions of diagnostic messages produced by Microsoft Windows Resource Compiler (RC). Many of these messages appear when RC is not able to compile resources properly. The descriptions in this topic clarify the causes. The messages are listed in alphabetic order.

A capital V in parentheses (V) at the beginning of a message description indicates that the message is displayed only if RC is run with the - V (verbose) option. These messages are generally informational and do not necessarily indicate errors.

### Accelerator Type required (ASCII or VIRTKEY)

The type parameter in the ACCELERATORS statement must contain either the ASCII or VIRTKEY value.

### BEGIN expected in Accelerator Table

An ACCELERATORS statement was not followed by the BEGIN keyword.

### BEGIN expected in Dialog

A DIALOG statement was not followed by the BEGIN keyword.

### BEGIN expected in menu

A MENU statement was not followed by the BEGIN keyword.

### BEGIN expected in RCData

An RCDATA statement was not followed by the BEGIN keyword.

### BEGIN expected in String Table

A STRINGTABLE statement was not followed by the BEGIN keyword.

### BEGIN expected in VERSIONINFO resource

A VERSIONINFO statement was not followed by the BEGIN keyword.

### Bitmap file resource-file is not in version-number format.

Use Microsoft Image Editor (IMAGEDIT.EXE) to convert version 2.x resource files to the version 3.1 format.

### Cannot Re-use String Constants

You are using the same value twice in a STRINGTABLE statement. Make sure that you have not mixed overlapping decimal and hexadecimal values.

### Control Character out of range [A - Z]

A control character in the ACCELERATORS statement is invalid. The character following the caret (^) must be in the range A through Z.

### Copying segment id (size bytes)

(V) Microsoft Windows Resource Compiler (RC) is copying the specified segment to the executable (.EXE) file.

### Could not find RCPP.EXE

The preprocessor (RCPP.EXE) must be in the current directory or in a directory specified in the PATH environment variable.

### Could not open in-file-name

Microsoft Windows Resource Compiler (RC) could not open the specified file. Make sure that the file exists and that you typed the filename correctly.

### Couldn't open resource-name

Microsoft Windows Resource Compiler (RC) could not open the specified file. Make sure that the file exists and that you typed the filename correctly.

### Creating resource-name

(V) Microsoft Windows Resource Compiler (RC) is creating a new binary resource (.RES) file.

### Empty menus not allowed

An END keyword appears before any menu items are defined in the MENU statement. Empty menus are not permitted by Microsoft Windows Resource Compiler (RC). Make sure that you do not have any opening quotation marks within the MENU statement.

### END expected in Dialog

The END keyword must appear at the end of a DIALOG statement. Make sure that there are no opening quotation marks left from the preceding statement.

### END expected in menu

The END keyword must appear at the end of a MENU statement. Make sure that there are no mismatched BEGIN and END statements.

### Error Creating resource-name

Microsoft Windows Resource Compiler (RC) could not create the specified binary resource (.RES)

file. Make sure that it is not being created on a read-only drive. Use the - V option to find out whether the file is being created.

Errors occurred when linking file.

The linker failed. For more information, see the documentation for your linker.

EXE file too large; relink with higher /ALIGN value

The executable (.EXE) file is too large. Relink the .EXE file with a larger value. For more information, see the documentation for your linker.

Expected Comma in Accelerator Table

Microsoft Windows Resource Compiler (RC) requires a comma between the event and idvalue parameters in the ACCELERATORS statement.

Expected control class name

The class parameter of a CONTROL statement in the DIALOG statement must be one of the following control types: BUTTON, COMBOBOX, EDIT, LISTBOX, SCROLLBAR, STATIC, or user-defined. Make sure that the class is spelled correctly.

Expected font face name

The typeface parameter of the FONT statement in the DIALOG statement must be an ASCII character string enclosed in double quotation marks. This parameter specifies the name of a font.

Expected ID value for MenuItem

The MENU statement must contain a MENUITEM statement, which has either an integer or a symbolic constant in the MenuID parameter.

Expected Menu String

Each MENUITEM and POPUP statement must contain a text parameter. This parameter is a string enclosed in double quotation marks that specifies the name of the menu item or pop-up menu. A MENUITEM SEPARATOR statement requires no quoted string.

Expected numeric command value

Microsoft Windows Resource Compiler (RC) was expecting a numeric idvalue parameter in the ACCELERATORS statement. Make sure that you have used a #define constant to specify the value and that the constant used is spelled correctly.

Expected numeric constant in string table

A numeric constant, defined in a #define statement, must immediately follow the BEGIN keyword in a STRINGTABLE statement.

Expected numeric point size

The pointsize parameter of the FONT statement in the DIALOG statement must be an integer point-size value.

Expected Numerical Dialog constant

A DIALOG statement requires integer values for the x, y, width, and height parameters. Make sure that these values, which are included after the DIALOG keyword, are not negative.

Expected String in STRINGTABLE

A string is expected after each numeric stringid parameter in a STRINGTABLE statement.

Expected String or Constant Accelerator command

Microsoft Windows Resource Compiler (RC) was not able to determine which key was being set up for the accelerator. The event parameter in the ACCELERATORS statement might be invalid.

Expected VALUE, BLOCK, or END keyword.

The VERSIONINFO structure requires a VALUE, BLOCK, or END keyword.

Expecting number for ID

A number is expected for the id parameter of a CONTROL statement in the DIALOG statement. Make sure that you have a number or a #define statement for the control identifier.

Expecting quoted string for key

The key string following the BLOCK or VALUE keyword should be enclosed in double quotation marks.

Expecting quoted string in dialog class

The class parameter of the CLASS statement in the DIALOG statement must be an integer or a string enclosed in double quotation marks.

Expecting quoted string in dialog title

The captiontext parameter of the CAPTION statement in the DIALOG statement must be an ASCII character string, enclosed in double quotation marks.

Fast-load area is [size] bytes at offset 0x[address]

(V) This is the size, in bytes, of all the following segments:

- Segments with the PRELOAD attribute
- Segments with the DISCARDABLE attribute

- Code segments that contain the entry point, WinMain
- Data segments (which should not be discardable)

To disable fast loading, use the `-k` option. Fast loading is the placement of segments in a contiguous area in the executable (.EXE) file for quicker loading. The offset is from the the beginning of the file.

File not created by LINK

You must create the executable (.EXE) file with an appropriate version of the linker.

File not found: filename

The file specified in the `rc` command was not found. Make sure that the file has not been moved to another directory and that the filename or path is typed correctly.

Font names must be ordinals

The `pointsize` parameter in the `FONT` statement must be an integer, not a string.

Insufficient memory to spawn RCPP.EXE

There wasn't enough memory to run the preprocessor (RCPP.EXE). Try disabling any memory-resident software that might be taking up too much memory. To verify the amount of memory you have, use the `chkdsk` command.

Invalid Accelerator

An event parameter in the ACCELERATORS statement was not recognized or was more than two characters long.

Invalid Accelerator Type (ASCII or VIRTKEY)

The type parameter in the ACCELERATORS statement must contain either the ASCII or VIRTKEY value.

Invalid control character

A control character in the ACCELERATORS statement is invalid. A valid control character consists of a caret (^) followed by a single letter.

Invalid Control type

The `CONTROL` statement in a `DIALOG` statement must be one of the following: CHECKBOX, COMBOBOX, CONTROL, CTEXT, DEFPUSHBUTTON, EDITTEXT, GROUPBOX, ICON, LISTBOX, LTEXT, PUSHBUTTON, RADIOBUTTON, RTEXT, or SCROLLBAR.

Invalid directive in preprocessed RC file

The specified filename has an embedded newline character.

Invalid .EXE file

The executable (.EXE) file is invalid. Make sure that the linker created it correctly and that the file exists.

Invalid switch, option

An option used was invalid. For a list of the command-line options, use the `rc -?` command.

Invalid type

The resource type was not among the types defined in the include file.

Invalid usage. Use `rc -?` for Help

Make sure that you have at least one filename to work with. For a list of the command-line options, use the `rc -?` command.

I/O error reading file.

Read failed. Since this is a generic routine, no specific filename is supplied.

I/O error seeking in file

Seeking in file failed. Since this is a generic routine, no specific filename is supplied.

I/O error writing file.

Write failed. Since this is a generic routine, no specific filename is supplied.

No executable filename specified.

The `-FE` option was used, but no executable (.EXE) file was specified.

No resource binary filename specified.

The `-FO` option was used, but no binary resource (.RES) file was specified.

Not a Microsoft Windows format .EXE file

Make sure that the linker created the executable (.EXE) file correctly and that the file exists.

Old DIB in resource-name. Pass it through `IMAGEDIT`.

The resource file specified is not compatible with Windows version 3.1. Make sure you have read and saved this file using the latest version of Microsoft Image Editor (`IMAGEDIT.EXE`). (Image Editor has replaced SDK Paint.)

Out of far heap memory

- There was not enough memory. Try disabling any memory-resident software that might be taking up too much space. To find out how much memory you have, use the `chkdsk` command.
- Out of memory, needed n bytes  
Microsoft Windows Resource Compiler (RC) was not able to allocate the specified amount of memory.
- RC: Invalid swap area size: -S string  
Invalid swap area size. Check your syntax for the `-S` option on the command line for Microsoft Windows Resource Compiler (RC). The following examples show acceptable command lines:
- ```
RC S123
RC S123K ; where K is kilobytes
RC S123p ; where p is paragraphs
```
- RC: Invalid switch: option  
An option used was invalid. For a list of the command-line options, use the `rc -?` command.
- RC: RCPP.ERR not found  
The RCPP.ERR file must be in the current directory or in a directory specified in the PATH environment variable.
- RC terminated by user  
A CTRL+C key combination was pressed, exiting Microsoft Windows Resource Compiler (RC).
- RC terminating after preprocessor errors  
For information about preprocessor errors, see the documentation for the preprocessor.
- RCPP.EXE command line greater than 128 bytes  
The command line for the preprocessor (RCPP.EXE) was too long.
- RCPP.EXE is not a valid executable  
The preprocessor (RCPP.EXE) may have been altered. Try copying the file again from the Microsoft Windows Software Development Kit (SDK) disks.
- Resource file resource-name is not in version-number format.  
Make sure your icons and cursors have been read and saved using the latest version of Microsoft Image Editor (IMAGEDIT.EXE).
- Resources will be aligned on number byte boundaries  
(V) The alignment is determined by an option on the command line for the linker.
- Sorting preload segments and resources into fast-load section  
(V) Microsoft Windows Resource Compiler (RC) is sorting the preloaded segments into a contiguous area in the executable (.EXE) file (the fast-load section) so that they can be loaded quickly.
- Text string or ordinal expected in Control  
The text parameter of a `CONTROL` statement in the `DIALOG` statement must be either a text string or an ordinal reference to the type of control that is expected. If using an ordinal, make sure that you have a `#define` statement for the control.
- The EXETYPE of this program is not Windows  
The `EXETYPE WINDOWS` statement did not appear in the module-definition (.DEF) file. Since the linker might make optimizations that are not appropriate for Windows, the `EXETYPE WINDOWS` statement must be specified.
- Unable to create destination  
Microsoft Windows Resource Compiler (RC) was not able to create the destination file. Make sure that there is enough disk space.
- Unable to open exe-file  
Microsoft Windows Resource Compiler (RC) could not open the executable (.EXE) file. Make sure that the linker created it correctly and that the file exists.
- Unbalanced Parentheses  
Make sure that you have closed every opening parenthesis in the `DIALOG` statement.
- Unexpected value in RCData  
The values for the raw-data parameter in the `RCDATA` statement must be integers or strings, separated by commas. Make sure that you did not leave out a comma or a quotation mark around a string.
- Unexpected value in value data  
A statement contained information with a format or size different from the expected value for that parameter.
- Unknown DIB header format

The device-independent bitmap (DIB) header is not a BITMAPCOREHEADER or BITMAPINFOHEADER structure.

Unknown error spawning RCPP.EXE

For an unknown reason, the preprocessor (RCPP.EXE) has not started. Try copying the file again from the SDK disks and use the `chkdsk` command to verify the amount of available memory.

Unknown Menu SubType

The item-definitions parameter of the MENU statement can contain only MENUITEM and POPUP statements.

Unrecognized VERSIONINFO field; BEGIN or comma expected

The format of the information following a VERSIONINFO statement is incorrect.

Version WORDs separated by commas expected

Values in an information block for a VERSIONINFO statement should be separated by commas.

Warning: ASCII character not equivalent to virtual key code

An invalid virtual-key code exists in the ACCELERATORS statement. The ASCII values for some characters (such as \*, ^, or &) are not equivalent to the virtual-key codes for the corresponding keys. (In the case of the asterisk [\*], the virtual-key code is equivalent to the ASCII value for 8, the numeric character on the same key. Therefore, the statement `VIRTKEY '* '` is invalid.)

Warning: SHIFT or CONTROL used without VIRTKEY

The `ALT`, `SHIFT`, and `CONTROL` options apply only to virtual keys in the ACCELERATORS statement. Make sure that the `VIRTKEY` option is used with one of these other options.

Warning: string segment number set to PRELOAD

Microsoft Windows Resource Compiler (RC) displays this warning when it copies a segment that must be preloaded but is not marked `PRELOAD` in the module-definition (`.DEF`) file for the linker. All nondiscardable segments should be preloaded, including automatic data segments, fixed segments, and the entry point of the code (WinMain). (The attributes of code segments are set by the `.DEF` file.)

Writing resource resource-name or ordinal-id. resource type (resource size)

(V) Microsoft Windows Resource Compiler (RC) is writing the resource name or ordinal identifier, followed by a period and the resource type and size, in bytes.

## ClassFirst (3.1)

```
#include toolhelp.h
BOOL ClassFirst(lpce)
CLASSENTRY FAR* lpce;          /* address of structure for class info */
```

The ClassFirst function fills the specified structure with general information about the first class in the Windows class list.

| Parameter | Description                                                                      |
|-----------|----------------------------------------------------------------------------------|
| lpce      | Points to a <u>CLASSENTRY</u> structure that will receive the class information. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The ClassFirst function can be used to begin a walk through the Windows class list. To examine subsequent items in the class list, an application can use the ClassNext function.

Before calling ClassFirst, an application must initialize the CLASSENTRY structure and specify its size, in bytes, in the dwSize member. An application can examine subsequent entries in the Windows class list by using the ClassNext function.

For more specific information about an individual class, use the GetClassInfo function, specifying the name of the class and instance handle from the CLASSENTRY structure.

### See Also

ClassNext, GetClassInfo, CLASSENTRY



### ClassNext (3.1)

#include toolhelp.h

BOOL ClassNext(lpce)

CLASSENTRY FAR\* lpce; /\* address of structure for class info \*/

The ClassNext function fills the specified structure with general information about the next class in the Windows class list.

| Parameter | Description                                                                      |
|-----------|----------------------------------------------------------------------------------|
| lpce      | Points to a <u>CLASSENTRY</u> structure that will receive the class information. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The ClassNext function can be used to continue a walk through the Windows class list started by the ClassFirst function.

For more specific information about an individual class, use the GetClassInfo function with the name of the class and instance handle from the CLASSENTRY structure.

#### See Also

ClassFirst, CLASSENTRY

## GlobalEntryHandle (3.1)

```
#include toolhelp.h
```

```
BOOL GlobalEntryHandle(lpge, hglb)
```

```
GLOBALENTRY FAR* lpge;          /* address of structure for object */
```

```
HGLOBAL hglb;
```

```
handle of item
```

```
/
```

```
/*
```

```
*
```

The GlobalEntryHandle function fills the specified structure with information that describes the given global memory object.

| Parameter | Description                                                                                        |
|-----------|----------------------------------------------------------------------------------------------------|
| lpge      | Points to a <u>GLOBALENTRY</u> structure that receives information about the global memory object. |
| hglb      | Identifies the global memory object to be described.                                               |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero. The function fails if the hglb value is an invalid handle or selector.

### Comments

This function retrieves information about a global memory handle or selector. Debuggers use this function to obtain the segment number of a segment loaded from an executable file.

Before calling the GlobalEntryHandle function, an application must initialize the GLOBALENTRY structure and specify its size, in bytes, in the dwSize member.

### See Also

GlobalEntryModule, GlobalFirst, GlobalInfo, GlobalNext, GLOBALENTRY

## GlobalEntryModule (3.1)

#include toolhelp.h

BOOL GlobalEntryModule(lpge, hmod, wSeg)

GLOBALENTRY FAR\* lpge; /\* address of structure for segment \*/

HMODULE hmod;

\* handle of module

/

WORD wSeg;

\* segment to describe

/

The GlobalEntryModule function fills the specified structure by lpge with information about the specified module segment.

| Parameter | Description                                                                                                                                                                                                                                                       |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpge      | Points to a <u>GLOBALENTRY</u> structure that receives information about the segment specified in the wSeg parameter.                                                                                                                                             |
| hmod      | Identifies the module that owns the segment.                                                                                                                                                                                                                      |
| wSeg      | Specifies the segment to be described in the <u>GLOBALENTRY</u> structure. The number of the first segment in the module is 1. Segment numbers are always contiguous, so if the last valid segment number is 10, all segment numbers 1 through 10 are also valid. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero. This function fails if the segment in the wSeg parameter does not exist in the module specified in the hmod parameter.

### Comments

Debuggers can use the GlobalEntryModule function to retrieve global heap information about a specific segment loaded from an executable file. Typically, the debugger will have symbols that refer to segment numbers; this function translates the segment numbers to heap information.

Before calling GlobalEntryModule, an application must initialize the GLOBALENTRY structure and specify its size, in bytes, in the dwSize member.

### See Also

GlobalEntryHandle, GlobalFirst, GlobalInfo, GlobalNext, GLOBALENTRY

### GlobalFirst (3.1)

#include toolhelp.h

BOOL GlobalFirst(lpge, wFlags)

GLOBALENTY FAR\* lpge;

WORD wFlags;

specifies the heap to use

/

/\* address of structure for object \*/

/\*

\*

The GlobalFirst function fills the specified structure with information that describes the first object on the global heap.

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |            |                                                                                                           |             |                                                                             |            |                                                                                                  |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|------------|-----------------------------------------------------------------------------------------------------------|-------------|-----------------------------------------------------------------------------|------------|--------------------------------------------------------------------------------------------------|
| lpge        | Points to a <u>GLOBALENTY</u> structure that receives information about the global memory object.                                                                                                                                                                                                                                                                                                                                                                                        |       |         |            |                                                                                                           |             |                                                                             |            |                                                                                                  |
| wFlags      | Specifies the heap to use. This parameter can be one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |            |                                                                                                           |             |                                                                             |            |                                                                                                  |
|             | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>GLOBAL_ALL</td><td>Structure pointed to by lpge will receive information about the first object on the complete global heap.</td></tr><tr><td>GLOBAL_FREE</td><td>Structure will receive information about the first object on the free list.</td></tr><tr><td>GLOBAL_LRU</td><td>Structure will receive information about the first object on the least-recently-used (LRU) list.</td></tr></tbody></table> | Value | Meaning | GLOBAL_ALL | Structure pointed to by lpge will receive information about the first object on the complete global heap. | GLOBAL_FREE | Structure will receive information about the first object on the free list. | GLOBAL_LRU | Structure will receive information about the first object on the least-recently-used (LRU) list. |
| Value       | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |            |                                                                                                           |             |                                                                             |            |                                                                                                  |
| GLOBAL_ALL  | Structure pointed to by lpge will receive information about the first object on the complete global heap.                                                                                                                                                                                                                                                                                                                                                                                |       |         |            |                                                                                                           |             |                                                                             |            |                                                                                                  |
| GLOBAL_FREE | Structure will receive information about the first object on the free list.                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |            |                                                                                                           |             |                                                                             |            |                                                                                                  |
| GLOBAL_LRU  | Structure will receive information about the first object on the least-recently-used (LRU) list.                                                                                                                                                                                                                                                                                                                                                                                         |       |         |            |                                                                                                           |             |                                                                             |            |                                                                                                  |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The GlobalFirst function can be used to begin a global heap walk. An application can examine subsequent objects on the global heap by using the GlobalNext function. Calls to GlobalNext must have the same wFlags value as that specified in GlobalFirst.

Before calling GlobalFirst, an application must initialize the GLOBALENTY structure and specify its size, in bytes, in the dwSize member.

#### See Also

GlobalEntryHandle, GlobalEntryModule, GlobalInfo, GlobalNext, GLOBALENTY

## GlobalHandleToSel (3.1)

#include toolhelp.h

WORD GlobalHandleToSel(hglb)

HGLOBAL hglb;

The GlobalHandleToSel function converts the given handle to a selector.

| Parameter | Description                                          |
|-----------|------------------------------------------------------|
| hglb      | Identifies the global memory object to be converted. |

### Returns

The return value is the selector of the given object if the function is successful. Otherwise, it is zero.

### Comments

The GlobalHandleToSel function converts a global handle to a selector appropriate for Windows, version 3.0 or 3.1, depending on which version is running. A debugging application might use this selector to access a global memory object if the object is not discardable or if the object's attributes are irrelevant.

### See Also

[GlobalAlloc](#)

### GlobalInfo function (3.1)

#include toolhelp.h

BOOL GlobalInfo(lpgi)

GLOBALINFO FAR\* lpgi; /\* address of global-heap structure \*/

The GlobalInfo function fills the specified structure with information that describes the global heap.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

|      |                                                                                          |
|------|------------------------------------------------------------------------------------------|
| lpgi | Points to a <u>GLOBALINFO</u> structure that receives information about the global heap. |
|------|------------------------------------------------------------------------------------------|

Returns

The return value is nonzero if the function successful. Otherwise, it is zero.

Comments

The information in the structure can be used to determine how much memory to allocate for a global heap walk.

Before calling the GlobalInfo function, an application must initialize the GLOBALINFO structure and specify its size, in bytes, in the dwSize member.

See Also

GlobalEntryHandle, GlobalEntryModule, GlobalFirst, GlobalNext, GLOBALINFO, GLOBALENTY

## GlobalNext (3.1)

#include toolhelp.h

BOOL GlobalNext(lpge, flags)

GLOBALENTY FAR\* lpge;

WORD flags;

to use

/

/\* address of structure for object \*/

/\* heap

\*

The GlobalNext function fills the specified structure with information that describes the next object on the global heap.

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |            |                                                                                                                      |             |                                                                             |            |                                                                                                  |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|------------|----------------------------------------------------------------------------------------------------------------------|-------------|-----------------------------------------------------------------------------|------------|--------------------------------------------------------------------------------------------------|
| lpge        | Points to a <u>GLOBALENTY</u> structure that receives information about the global memory object.                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |            |                                                                                                                      |             |                                                                             |            |                                                                                                  |
| flags       | Specifies heap to use. This parameter can be one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |            |                                                                                                                      |             |                                                                             |            |                                                                                                  |
|             | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>GLOBAL_ALL</td><td>Structure pointed by the lpge parameter will receive information about the first object on the complete global heap.</td></tr><tr><td>GLOBAL_FREE</td><td>Structure will receive information about the first object on the free list.</td></tr><tr><td>GLOBAL_LRU</td><td>Structure will receive information about the first object on the least-recently-used (LRU) list.</td></tr></tbody></table> | Value | Meaning | GLOBAL_ALL | Structure pointed by the lpge parameter will receive information about the first object on the complete global heap. | GLOBAL_FREE | Structure will receive information about the first object on the free list. | GLOBAL_LRU | Structure will receive information about the first object on the least-recently-used (LRU) list. |
| Value       | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |            |                                                                                                                      |             |                                                                             |            |                                                                                                  |
| GLOBAL_ALL  | Structure pointed by the lpge parameter will receive information about the first object on the complete global heap.                                                                                                                                                                                                                                                                                                                                                                                |       |         |            |                                                                                                                      |             |                                                                             |            |                                                                                                  |
| GLOBAL_FREE | Structure will receive information about the first object on the free list.                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |            |                                                                                                                      |             |                                                                             |            |                                                                                                  |
| GLOBAL_LRU  | Structure will receive information about the first object on the least-recently-used (LRU) list.                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |            |                                                                                                                      |             |                                                                             |            |                                                                                                  |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The GlobalNext function can be used to continue a global heap walk started by the GlobalFirst, GlobalEntryHandle, or GlobalEntryModule functions.

If GlobalFirst starts a heap walk, the flags value used in GlobalNext must be the same as the value used in GlobalFirst.

### See Also

GlobalEntryHandle, GlobalEntryModule, GlobalFirst, GlobalInfo, GLOBALENTY

### InterruptRegister (3.1)

```
#include toolhelp.h
```

```
BOOL InterruptRegister(htask, lpfnIntCallback)
```

```
HTASK htask; /* handle of task */
```

```
FARPROC lpfnIntCallback;
```

```
* address of callback function
```

```
/
```

The `InterruptRegister` function installs a callback function to handle all system interrupts.

| Parameter       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| htask           | Identifies the task that is registering the callback function. The htask value is for registration purposes, not for filtering interrupts. Typically, this value is NULL, indicating the current task. The only time this value is not NULL is when an application requires more than one interrupt handler.                                                                                                                                                                         |
| lpfnIntCallback | Points to the interrupt callback function that will handle interrupts. The Tool Helper library calls this function whenever a task receives an interrupt.<br>The lpfnIntCallback value is normally the return value of a call to the <code>MakeProcInstance</code> function. This causes the interrupt callback function to be entered with the AX register set to the selector of the application's data segment. Usually, an exported function prolog contains the following code: |

```
mov ds,ax
```

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The syntax of the function pointed to by lpfnIntCallback is as follows:

```
void InterruptRegisterCallback(void)
```

`InterruptRegisterCallback` is a placeholder for the application-defined function name. The actual name must be exported by including it an EXPORTS in the application's module-definition file.

An interrupt callback function must be reentrant, must be page-locked, and must explicitly preserve all register values.

|                   |  |          |
|-------------------|--|----------|
| ·                 |  |          |
| ·                 |  |          |
| ·                 |  |          |
| -----             |  |          |
| SS (fault)        |  | SP + 12h |
| -----             |  |          |
| SP (fault)        |  | SP + 10h |
| -----             |  |          |
| Flags (fault)     |  | SP + 0Eh |
| -----             |  |          |
| CS (fault)        |  | SP + 0Ch |
| -----             |  |          |
| IP (fault)        |  | SP + 0Ah |
| -----             |  |          |
| handle (internal) |  | SP + 08h |
| -----             |  |          |
| interrupt number  |  | SP + 06h |
| -----             |  |          |
| AX                |  | SP + 04h |
| -----             |  |          |
| CS (toolhelp.dll) |  | SP + 02h |
| -----             |  |          |
| IP (toolhelp.dll) |  | SP + 00h |
| +-----+           |  |          |



The SS and SP values will not be on the stack unless a low-stack fault occurred. This fault is indicated by the high bit of the interrupt number being set.

When Windows calls a callback function, the AX register contains the DS value for the instance of the application that contains the callback function. For more information about this process, see the [MakeProcInstance](#) function.

Typically, an interrupt callback function is exported. If it is not exported, the developer should verify that the appropriate stack frame is generated, including the correct DS value.

An interrupt callback function must save and restore all register values. The function must also do one of the following:

- Execute an `retf` instruction if it does not handle the interrupt. The Tool Helper library will pass the interrupt to the next appropriate handler in the interrupt handler list.
- Terminate the application by using the [TerminateApp](#) function.
- Correct the problem that caused the interrupt, clear the first 10 bytes of the stack, and execute an `iret` instruction. This action will restart execution at the specified address. An application may change this address, if necessary.
- Execute a nonlocal goto to a known position in the application by using the [Catch and Throw](#) functions. This type of interrupt handling can be hazardous; the system may be in an unstable state and another fault may occur. Applications that handle interrupts in this way must verify that the fault was a result of the application's code.

The Tool Helper library supports the following interrupts:

| Name             | Number | Meaning                                                  |
|------------------|--------|----------------------------------------------------------|
| INT_DIV0         | 0      | Divide-error exception                                   |
| INT_1            | 1      | Debugger interrupt                                       |
| INT_3            | 3      | Breakpoint interrupt                                     |
| INT_UDINSTR      | 6      | Invalid-opcode exception                                 |
| INT_STKFAULT     | 12     | Stack exception                                          |
| INT_GPFALT       | 13     | General protection violation                             |
| INT_BADPAGEFAULT | 14     | Page fault not caused by normal virtual-memory operation |
| INT_CTLALTSYS RQ | 256    | User pressed CTRL+ALT+SYS RQ                             |

The Tool Helper library returns interrupt numbers as word values. Normal software interrupts and processor faults are represented by numbers in the range 0 through 255. Interrupts specific to Tool Helper are represented by numbers greater than 255.

Some developers may wish to use CTRL+ALT+SYS RQ (Interrupt 256) to break into the debugger. Be cautious about implementing this interrupt, because the point at which execution stops will probably be in a sensitive part of the Windows kernel. All [InterruptRegisterCallback](#) functions must be page-locked to prevent problems when this interrupt is used. In addition, the debugger probably will not be able to perform user-interface functions. However, the debugger can use Tool Helper functions to set breakpoints and gather information. The debugger may also be able to use a debugging terminal or secondary screen to display information.

#### Low-stack Faults

A low-stack fault occurs when inadequate stack space is available on the faulting application's stack. For example, if any fault occurs when there is less than 128 bytes of stack space available or if runaway recursion depletes the stack, a low-stack fault occurs. The Tool Helper library processes a low-stack fault differently than it processes other faults.

A low-stack fault is indicated by the high-order bit of the interrupt number being set. For example, if a stack fault occurs and the SP value becomes invalid, the Tool Helper library will return the fault number as 0x800C rather than 0x000C.

Interrupt handlers designed to process low-stack faults must be aware that the Tool Helper library has passed a fault frame on a stack other than the faulting application's stack. The SS:SP value is on the stack because it was pushed before the rest of the information in the stack frame. The SS:SP value is available only for advisory purposes.

An interrupt handler should never restart the faulting instruction, because this will cause the system to crash. The handler may terminate the application with [TerminateApp](#) or pass the fault to the next handler in the interrupt-handler list.

Interrupt handlers should not assume that all stack faults are low-stack faults. For example, if an application accesses a stack-relative variable that is out of range, a stack fault will occur. This type of fault can be processed in the same manner as any general protection (GP) fault. If the high-order bit of the interrupt number is not set, the instruction can be restarted.

Interrupt handlers also should not assume that all low-stack faults are stack faults. Any fault that occurs when there is less than 128 bytes of stack available will cause a low-stack fault.

Interrupt callback functions that are not designed to process low-stack faults should execute an `retf` instruction so that the Tool Helper library will pass the fault to the next appropriate handler in the interrupt-handler list.

See Also

[Catch](#), [InterruptUnRegister](#), [NotifyRegister](#), [NotifyUnRegister](#), [TerminateApp](#), [Throw](#)

## InterruptUnRegister (3.1)

#include toolhelp.h

```
BOOL InterruptUnRegister(htask)
HTASK htask;          /* handle of task */
```

The InterruptUnRegister function restores the default interrupt handle for system interrupts.

| Parameter | Description                                                                 |
|-----------|-----------------------------------------------------------------------------|
| htask     | Identifies the task. If this value is NULL, it identifies the current task. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

After this function is executed, the Tool Helper library will pass all interrupts it receives to the system's default interrupt handler.

### See Also

[InterruptRegister](#), [NotifyRegister](#), [NotifyUnRegister](#), [TerminateApp](#)

### LocalFirst (3.1)

```
#include toolhelp.h
```

```
BOOL LocalFirst(lple, hglbHeap)
```

```
LOCALENTRY FAR* lple;          /* address of LOCALENTRY structure */
```

```
HGLOBAL hglbHeap;
```

```
/* handle of local heap
```

```
/
```

The LocalFirst function fills the specified structure with information that describes the first object on the local heap.

| Parameter | Description                                                                                          |
|-----------|------------------------------------------------------------------------------------------------------|
| lple      | Points to a <u>LOCALENTRY</u> structure that will receive information about the local memory object. |
| hglbHeap  | Identifies the local heap.                                                                           |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The LocalFirst function can be used to begin a local heap walk. An application can examine subsequent objects on the local heap by using the LocalNext function.

Before calling LocalFirst, an application must initialize the LOCALENTRY structure and specify its size, in bytes, in the dwSize member.

#### See Also

LocalInfo, LocalNext, LOCALENTRY

### LocalInfo function (3.1)

#include toolhelp.h

BOOL LocalInfo(lpli, hglbHeap)

LOCALINFO FAR\* lpli; /\* address of LOCALINFO structure \*/

HGLOBAL hglbHeap;

\* handle of local heap

/

The LocalInfo function fills the specified structure with information that describes the local heap.

| Parameter | Description                                                                         |
|-----------|-------------------------------------------------------------------------------------|
| lpli      | Points to a LOCALINFO structure that will receive information about the local heap. |
| hglbHeap  | Identifies the local heap to be described.                                          |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The information in the LOCALINFO structure can be used to determine how much memory to allocate for a local heap walk.

Before calling LocalInfo, an application must initialize the LOCALINFO structure and specify its size, in bytes, in the dwSize member.

#### See Also

LocalFirst, LocalNext, LOCALINFO, LOCALENTRY

### LocalNext (3.1)

#include toolhelp.h

BOOL LocalNext(lple)

LOCALENTRY FAR\* lple; /\* address of LOCALENTRY structure \*/

The LocalNext function fills the specified structure with information that describes the next object on the local heap.

| Parameter | Description                                                                                          |
|-----------|------------------------------------------------------------------------------------------------------|
| lple      | Points to a <u>LOCALENTRY</u> structure that will receive information about the local memory object. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The LocalNext function can be used to continue a local heap walk started by the LocalFirst function.

#### See Also

LocalFirst, LocalInfo, LOCALENTRY

### MemManInfo function (3.1)

#include toolhelp.h

BOOL MemManInfo(lpmmi)

MEMMANINFO FAR\* lpmmi; /\* address of MEMMANINFO structure \*/

The MemManInfo function fills the specified structure with status and performance information about the memory manager. This function is most useful in 386 enhanced mode but can also be used in standard mode.

| Parameter | Description                                                                                     |
|-----------|-------------------------------------------------------------------------------------------------|
| lpmmi     | Points to a <u>MEMMANINFO</u> structure that will receive information about the memory manager. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

This function is included for advisory purposes.

Before calling MemManInfo, an application must initialize the MEMMANINFO structure and specify its size, in bytes, in the dwSize member.

#### See Also

MEMMANINFO

## MemoryRead (3.1)

#include toolhelp.h

```
DWORD MemoryRead(wSel, dwOffset, lpvBuf, dwcb)
WORD wSel;          /* selector of global heap object */
DWORD dwOffset;    /* offset to
object */
/
void FAR* lpvBuf;
* address of buffer to read to
/
DWORD dwcb;
* number of bytes to read
/
```

The MemoryRead function copies memory from the specified global heap object to the specified buffer.

| Parameter | Description                                                                                                                                                                                                                                                                                            |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wSel      | Specifies the global heap object from which to read. This value must be a selector on the global heap; if the value is an alias selector or a selector in a tiled selector array, MemoryRead will fail.                                                                                                |
| dwOffset  | Specifies the offset in the object specified in the wSel parameter at which to begin reading. The dwOffset value may point anywhere within the object; it may be greater than 64K if the object is larger than 64K.                                                                                    |
| lpvBuf    | Points to the buffer to which MemoryRead will copy the memory from the object. This buffer must be large enough to contain the entire amount of memory copied to it. If the application is running under low memory conditions, lpvBuf should be in a fixed object while MemoryRead copies data to it. |
| dwcb      | Specifies the number of bytes to copy from the object to the buffer pointed to by lpvBuf.                                                                                                                                                                                                              |

### Returns

The return value is the number of bytes copied from wSel to lpvBuf. If wSel is invalid or if dwOffset is out of the selector's range, the return value is zero.

### Comments

The MemoryRead function enables developers to examine memory without consideration for selector tiling and aliasing. MemoryRead reads memory in read-write or read-only objects. This function can be used in any size object owned by any task. It is not necessary to compute selector array offsets.

The MemoryRead and MemoryWrite functions are designed to read and write objects loaded by the LoadModule function or allocated by the GlobalAlloc function. Developers should not split off the selector portion of a far pointer and use this as the value for wSel, unless the selector is known to be on the global heap.

### See Also

MemoryWrite



### MemoryWrite (3.1)

```
#include toolhelp.h
```

```
DWORD MemoryWrite(wSel, dwOffset, lpvBuf, dwcb)
WORD wSel;          /* selector of global heap object */
DWORD dwOffset;    /* offset to object */
void FAR* lpvBuf;  /* address of buffer to write from */
DWORD dwcb;       /* number of bytes to write */
```

The MemoryWrite function copies memory from the specified buffer to the specified global heap object.

| Parameter | Description                                                                                                                                                                                                           |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wSel      | Specifies the global heap object to which MemoryWrite will write. This value must be a selector on the global heap; if the value is an alias selector or a selector in a tiled selector array, MemoryWrite will fail. |
| dwOffset  | Specifies the offset in the object at which to begin writing. The dwOffset value may point anywhere within the object; it may be greater than 64K if the object is larger than 64K.                                   |
| lpvBuf    | Points to the buffer from which MemoryWrite will copy the memory to the object. If the application is running under low memory conditions, lpvBuf should be in a fixed object while MemoryWrite copies data from it.  |
| dwcb      | Specifies the number of bytes to copy to the object from the buffer pointed to by lpvBuf.                                                                                                                             |

#### Returns

The return value is the number of bytes copied from lpvBuf to wSel. If the selector is invalid or if dwOffset is out of the selector's range, the return value is zero.

#### Comments

The MemoryWrite function enables developers to modify memory without consideration for selector tiling and aliasing. MemoryWrite writes memory in read-write or read-only objects. This function can be used in any size object owned by any task. It is not necessary to make alias objects writable or to compute selector array offsets.

The MemoryRead and MemoryWrite functions are designed to read and write objects loaded by the LoadModule function or allocated by the GlobalAlloc function. Developers should not split off the selector portion of a far pointer and use this as the value for wSel, unless the selector is known to be on the global heap.

See Also

MemoryRead

### ModuleFindHandle (3.1)

```
#include toolhelp.h
```

```
HMODULE ModuleFindHandle(lpme, hmod)
```

```
MODULEENTRY FAR* lpme;          /* address of MODULEENTRY structure */
```

```
HMODULE hmod;
```

```
/* handle of module
```

```
/
```

The ModuleFindHandle function fills the specified structure with information that describes the given module.

| Parameter | Description                                                                              |
|-----------|------------------------------------------------------------------------------------------|
| lpme      | Points to a <u>MODULEENTRY</u> structure that will receive information about the module. |
| hmod      | Identifies the module to be described.                                                   |

#### Returns

The return value is the handle of the given module if the function is successful. Otherwise, it is NULL.

#### Comments

The ModuleFindHandle function returns information about a currently loaded module whose module handle is known.

This function can be used to begin a walk through the list of all currently loaded modules. An application can examine subsequent items in the module list by using the ModuleNext function.

Before calling ModuleFindHandle, an application must initialize the MODULEENTRY structure and specify its size, in bytes, in the dwSize member.

#### See Also

ModuleFindName, ModuleFirst, ModuleNext, MODULEENTRY

## ModuleFindName (3.1)

#include toolhelp.h

```
HMODULE ModuleFindName(lpme, lpszName)
MODULEENTRY FAR* lpme;          /* address of MODULEENTRY structure */
LPCSTR lpszName;
/* address of module name
/
```

The ModuleFindName function fills the specified structure with information that describes the module with the specified name.

| Parameter | Description                                                                              |
|-----------|------------------------------------------------------------------------------------------|
| lpme      | Points to a <u>MODULEENTRY</u> structure that will receive information about the module. |
| lpszName  | Specifies the name of the module to be described.                                        |

### Returns

The return value is the handle named in the lpszName parameter, if the function is successful. Otherwise, it is NULL.

### Comments

The ModuleFindName function returns information about a currently loaded module by looking up the module's name in the module list.

This function can be used to begin a walk through the list of all currently loaded modules. An application can examine subsequent items in the module list by using the ModuleNext function.

Before calling ModuleFindName, an application must initialize the MODULEENTRY structure and specify its size, in bytes, in the dwSize member.

### See Also

ModuleFindHandle, ModuleFirst, ModuleNext, MODULEENTRY

### ModuleFirst (3.1)

```
#include toolhelp.h
```

```
BOOL ModuleFirst(lpme)
```

```
MODULEENTRY FAR* lpme;          /* address of MODULEENTRY structure */
```

The ModuleFirst function fills the specified structure with information that describes the first module in the list of all currently loaded modules.

| Parameter | Description                                                                                    |
|-----------|------------------------------------------------------------------------------------------------|
| lpme      | Points to a <u>MODULEENTRY</u> structure that will receive information about the first module. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The ModuleFirst function can be used to begin a walk through the list of all currently loaded modules. An application can examine subsequent items in the module list by using the ModuleNext function.

Before calling ModuleFirst, an application must initialize the MODULEENTRY structure and specify its size, in bytes, in the dwSize member.

#### See Also

ModuleFindHandle, ModuleFindName, ModuleNext, MODULEENTRY

### ModuleNext (3.1)

```
#include toolhelp.h
```

```
BOOL ModuleNext(lpme)
```

```
MODULEENTRY FAR* lpme;          /* address of MODULEENTRY structure */
```

The ModuleNext function fills the specified structure with information that describes the next module in the list of all currently loaded modules.

| Parameter | Description                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------|
| lpme      | Points to a <u>MODULEENTRY</u> structure that will receive information about the next module. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The ModuleNext function can be used to continue a walk through the list of all currently loaded modules. The walk must have been started by the ModuleFirst, ModuleFindName, or ModuleFindHandle function.

#### See Also

ModuleFindHandle, ModuleFindName, ModuleFirst, MODULEENTRY

## NotifyRegister (3.1)

```
#include toolhelp.h
```

```
BOOL NotifyRegister(htask, lpfnCallback, wFlags)
HTASK htask; /* handle of task */
LPFNNOTIFYCALLBACK lpfnCallback;
/* address of callback function
/
WORD wFlags;
* notification flags
/
```

The NotifyRegister function installs a notification callback function for the given task.

| Parameter     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |           |                                                                                                                                                    |               |                                                                                                                                                                          |        |                                                                        |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|------------------------------------------------------------------------|
| htask         | Identifies the task associated with the callback function. If this parameter is NULL, it identifies the current task.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |           |                                                                                                                                                    |               |                                                                                                                                                                          |        |                                                                        |
| lpfnCallback  | Points to the notification callback function that is installed for the task. The kernel calls this function whenever it sends a notification to the task.<br>The callback-function address is normally the return value of a call to <u>MakeProcInstance</u> . This causes the callback function to be entered with the AX register set to the selector of the application's data segment. Usually, an exported function prolog contains the following code:<br><pre>mov ds, ax</pre>                                                                                                                                                                                                                                                                                                              |       |         |           |                                                                                                                                                    |               |                                                                                                                                                                          |        |                                                                        |
| wFlags        | Specifies the optional notifications that the application will receive, in addition to the default notifications. This parameter can be NF_NORMAL or any combination of the following values:<br><table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>NF_NORMAL</td><td>The application will receive the default notifications but none of the notifications of task switching, system debugging errors, or debug strings.</td></tr><tr><td>NF_TASKSWITCH</td><td>The application will receive task-switching notifications. To avoid poor performance, an application should not receive these notifications unless absolutely necessary.</td></tr><tr><td>NF_RIP</td><td>The application will receive notifications of system debugging errors.</td></tr></tbody></table> | Value | Meaning | NF_NORMAL | The application will receive the default notifications but none of the notifications of task switching, system debugging errors, or debug strings. | NF_TASKSWITCH | The application will receive task-switching notifications. To avoid poor performance, an application should not receive these notifications unless absolutely necessary. | NF_RIP | The application will receive notifications of system debugging errors. |
| Value         | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |           |                                                                                                                                                    |               |                                                                                                                                                                          |        |                                                                        |
| NF_NORMAL     | The application will receive the default notifications but none of the notifications of task switching, system debugging errors, or debug strings.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |           |                                                                                                                                                    |               |                                                                                                                                                                          |        |                                                                        |
| NF_TASKSWITCH | The application will receive task-switching notifications. To avoid poor performance, an application should not receive these notifications unless absolutely necessary.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |           |                                                                                                                                                    |               |                                                                                                                                                                          |        |                                                                        |
| NF_RIP        | The application will receive notifications of system debugging errors.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |           |                                                                                                                                                    |               |                                                                                                                                                                          |        |                                                                        |

### Returns

The return value is nonzero if the function was successful. Otherwise, it is zero.

### Callback Function

The syntax of the function pointed to by lpfnCallback is as follows:

```
BOOL NotifyRegisterCallback(wID, dwData)
```

```
WORD wID;
```

```
DWORD dwData;
```

| Parameter     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |               |                                                                       |              |                                                              |             |                                                                          |            |                                                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---------------|-----------------------------------------------------------------------|--------------|--------------------------------------------------------------|-------------|--------------------------------------------------------------------------|------------|--------------------------------------------------------------------------------------------------------------------------------------|
| wID           | Indicates the type of notification and the value of the dwData parameter. The wID parameter may be one of the following values in Windows versions 3.0 and later:<br><table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>NFY_DELMODULE</td><td>The low-order word of dwData is the handle of the module to be freed.</td></tr><tr><td>NFY_EXITTASK</td><td>The low-order byte of dwData contains the program exit code.</td></tr><tr><td>NFY_FREESEG</td><td>The low-order word of dwData is the selector of the segment to be freed.</td></tr><tr><td>NFY_INCHAR</td><td>The dwData parameter is not used. The notification callback function should return either the ASCII value for the keystroke or NULL.</td></tr></tbody></table> | Value | Meaning | NFY_DELMODULE | The low-order word of dwData is the handle of the module to be freed. | NFY_EXITTASK | The low-order byte of dwData contains the program exit code. | NFY_FREESEG | The low-order word of dwData is the selector of the segment to be freed. | NFY_INCHAR | The dwData parameter is not used. The notification callback function should return either the ASCII value for the keystroke or NULL. |
| Value         | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |               |                                                                       |              |                                                              |             |                                                                          |            |                                                                                                                                      |
| NFY_DELMODULE | The low-order word of dwData is the handle of the module to be freed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |               |                                                                       |              |                                                              |             |                                                                          |            |                                                                                                                                      |
| NFY_EXITTASK  | The low-order byte of dwData contains the program exit code.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |               |                                                                       |              |                                                              |             |                                                                          |            |                                                                                                                                      |
| NFY_FREESEG   | The low-order word of dwData is the selector of the segment to be freed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |               |                                                                       |              |                                                              |             |                                                                          |            |                                                                                                                                      |
| NFY_INCHAR    | The dwData parameter is not used. The notification callback function should return either the ASCII value for the keystroke or NULL.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |               |                                                                       |              |                                                              |             |                                                                          |            |                                                                                                                                      |

|                                                                 |                                                                                                          |
|-----------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| NFY_LOADSEG                                                     | The dwData parameter points to an <u>NFYLOADSEG</u> structure.                                           |
| NFY_OUTSTR                                                      | The dwData parameter points to the string to be displayed.                                               |
| NFY_STARTDLL                                                    | The dwData parameter points to an <u>NFYSTARTDLL</u> structure.                                          |
| NFY_STARTTASK                                                   | The dwData parameter is the CS:IP of the starting address of the task.                                   |
| NFY_UNKNOWN                                                     | The kernel returned an unknown notification.                                                             |
| In Windows version 3.1, wID may be one of the following values: |                                                                                                          |
| Value                                                           | Meaning                                                                                                  |
| NFY_LOGERROR                                                    | The dwData parameter points to an <u>NFYLOGERROR</u> structure.                                          |
| NFY_LOGPARAMERROR                                               | The dwData parameter points to an <u>NFYLOGPARAMERROR</u> structure.                                     |
| NFY_RIP                                                         | The dwData parameter points to an <u>NFYRIP</u> structure.                                               |
| NFY_TASKIN                                                      | The dwData parameter is undefined. The callback function should call the <u>GetCurrentTask</u> function. |
| NFY_TASKOUT                                                     | The dwData parameter is undefined. The callback function should call <u>GetCurrentTask</u> .             |

dwData Specifies data, or specifies a pointer to data, or is undefined, depending on the value of wID.

#### Returns

The return value of the callback function is nonzero if the callback function handled the notification. Otherwise, it is zero and the notification is passed to other callback functions.

#### Comments

A notification callback function must be able to ignore any unknown notification value. Typically, the notification callback function cannot use any Windows function, with the exception of the Tool Helper functions and PostMessage.

NotifyRegisterCallback is a placeholder for the application-defined function name. The actual name must be exported by including it in an EXPORTS statement in the application's module-definition file.

#### See Also

InterruptRegister, InterruptUnRegister, MakeProcInstance, NotifyUnRegister, TerminateApp, NFYLOADSEG, NFYLOGERROR, NFYLOGPARAMERROR, NFYRIP, NFYSTARTDLL

## NotifyUnRegister (3.1)

#include toolhelp.h

BOOL NotifyUnRegister(htask)

HTASK htask; /\* handle of task \*/

The NotifyUnRegister function restores the default notification handler.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

|       |                                                                        |
|-------|------------------------------------------------------------------------|
| htask | Identifies the task. If htask is NULL, it identifies the current task. |
|-------|------------------------------------------------------------------------|

Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

Comments

After this function is executed, the given task no longer receives notifications from the kernel.

See Also

[InterruptRegister](#), [InterruptUnRegister](#), [NotifyRegister](#), [TerminateApp](#)



### StackTraceCSIPFirst (3.1)

```
#include toolhelp.h
```

```
BOOL StackTraceCSIPFirst(lpste, wSS, wCS, wIP, wBP)
```

```
STACKTRACEENTRY FAR* lpste; /* address of stack-frame structure */
```

```
WORD wSS;
```

```
* value of SS register
```

```
/
```

```
WORD wCS;
```

```
* value of CS register
```

```
/
```

```
WORD wIP;
```

```
* value of IP register
```

```
/
```

```
WORD wBP;
```

```
* value of BP register
```

```
/
```

The StackTraceCSIPFirst function fills the specified structure with information that describes the specified stack frame.

| Parameter | Description                                                                                                                  |
|-----------|------------------------------------------------------------------------------------------------------------------------------|
| lpste     | Points to a <u>STACKTRACEENTRY</u> structure to receive information about the stack.                                         |
| wSS       | Contains the value in the SS register. This value is used with the wBP value to determine the next entry in the stack trace. |
| wCS       | Contains the value in the CS register of the first stack frame.                                                              |
| wIP       | Contains the value in the IP register of the first stack frame.                                                              |
| wBP       | Contains the value in the BP register. This value is used with the wSS value to determine the next entry in the stack trace. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The StackTraceFirst function can be used to begin a stack trace of any task except the current task. When a task is inactive, the kernel maintains its state, including its current stack, stack pointer, CS and IP values, and BP value. The kernel does not maintain these values for the current task. Therefore, when a stack trace is done on the current task, the application must use StackTraceCSIPFirst to begin a stack trace. An application can continue to trace through the stack by using the StackTraceNext function.

Before calling StackTraceCSIPFirst, an application must initialize the STACKTRACEENTRY structure and specify its size, in bytes, in the dwSize member.

#### See Also

StackTraceNext, StackTraceFirst, STACKTRACEENTRY

### StackTraceFirst (3.1)

#include toolhelp.h

BOOL StackTraceFirst(lpste, htask)

STACKTRACEENTRY FAR\* lpste;

/\* address of stack-frame structure \*/

HTASK htask;

\* handle of task

/

The StackTraceFirst function fills the specified structure with information that describes the first stack frame for the given task.

| Parameter | Description                                                                                             |
|-----------|---------------------------------------------------------------------------------------------------------|
| lpste     | Points to a <u>STACKTRACEENTRY</u> structure to receive information about the task's first stack frame. |
| htask     | Identifies the task whose stack information is to be described.                                         |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The StackTraceFirst function can be used to begin a stack trace of any task except the current task. When a task is inactive, the kernel maintains its state, including its current stack, stack pointer, CS and IP values, and BP value. The kernel does not maintain these values for the current task. Therefore, when a stack trace is done on the current task, the application must use the StackTraceCSIPFirst function to begin a stack trace. An application can continue to trace through the stack by using the StackTraceNext function.

Before calling StackTraceFirst, an application must initialize the STACKTRACEENTRY structure and specify its size, in bytes, in the dwSize member.

#### See Also

StackTraceCSIPFirst, StackTraceNext, STACKTRACEENTRY

### StackTraceNext (3.1)

#include toolhelp.h

BOOL StackTraceNext(lpste)

STACKTRACEENTRY FAR\* lpste; /\* address of stack-frame structure \*/

The StackTraceNext function fills the specified structure with information that describes the next stack frame in a stack trace.

| Parameter | Description                                                                                     |
|-----------|-------------------------------------------------------------------------------------------------|
| lpste     | Points to a <u>STACKTRACEENTRY</u> structure to receive information about the next stack frame. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The StackTraceNext function can be used to continue a stack trace started by using the StackTraceFirst or StackTraceCSIPFirst function.

#### See Also

StackTraceCSIPFirst, StackTraceFirst, STACKTRACEENTRY

## SystemHeapInfo (3.1)

#include toolhelp.h

BOOL SystemHeapInfo(lpshi)

SYSHEAPINFO FAR\* lpshi; /\* address of heap-info structure \*/

The SystemHeapInfo function fills the specified structure with information that describes the USER.EXE and GDI.EXE heaps.

| Parameter | Description                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------|
| lpshi     | Points to a <u>SYSHEAPINFO</u> structure to receive information about the USER and GDI heaps. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

This function is included for advisory purposes. Before calling SystemHeapInfo, an application must initialize the SYSHEAPINFO structure and specify its size, in bytes, in the dwSize member.

### See Also

SYSHEAPINFO

## TaskFindHandle (3.1)

#include toolhelp.h

BOOL TaskFindHandle(lpte, htask)

TASKENTRY FAR\* lpte; /\* address of TASKENTRY structure \*/

HTASK htask; /\*

handle of task

/

\*

The TaskFindHandle function fills the specified structure with information that describes the given task.

| Parameter | Description                                                                   |
|-----------|-------------------------------------------------------------------------------|
| lpte      | Points to a <u>TASKENTRY</u> structure to receive information about the task. |
| htask     | Identifies the task to be described.                                          |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The TaskFindHandle function can be used to begin a walk through the task queue. An application can examine subsequent entries in the task queue by using the TaskNext function.

Before calling TaskFindHandle, an application must initialize the TASKENTRY structure and specify its size, in bytes, in the dwSize member.

### See Also

TaskFirst, TaskNext, TASKENTRY

### TaskFirst (3.1)

#include toolhelp.h

BOOL TaskFirst(lpte)

TASKENTRY FAR\* lpte; /\* address of TASKENTRY structure \*/

The TaskFirst function fills the specified structure with information about the first task on the task queue.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

|      |                                                                                     |
|------|-------------------------------------------------------------------------------------|
| lpte | Points to a <u>TASKENTRY</u> structure to receive information about the first task. |
|------|-------------------------------------------------------------------------------------|

Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

Comments

The TaskFirst function can be used to begin a walk through the task queue. An application can examine subsequent entries in the task queue by using the TaskNext function.

Before calling TaskFirst, an application must initialize the TASKENTRY structure and specify its size, in bytes, in the dwSize member.

See Also

TaskFindHandle, TaskNext, TASKENTRY

## TaskGetCSIP (3.1)

#include toolhelp.h

DWORD TaskGetCSIP(htask)

HTASK htask; /\* handle of task \*/

The TaskGetCSIP function returns the next CS:IP value of a sleeping task. This function is useful for applications that must "know" where a sleeping task will begin execution upon awakening.

| Parameter | Description                                                                                                                 |
|-----------|-----------------------------------------------------------------------------------------------------------------------------|
| htask     | Identifies the task whose CS:IP value is being examined. This task must be sleeping when the application calls TaskGetCSIP. |

### Returns

The return value is the next CS:IP value, if the function is successful. If the htask parameter is invalid, the return value is NULL.

### Comments

TaskGetCSIP should not be called if htask identifies the current task.

### See Also

[DirectedYield](#), [TaskSetCSIP](#), [TaskSwitch](#)

## TaskNext (3.1)

#include toolhelp.h

BOOL TaskNext(lpte)

TASKENTRY FAR\* lpte; /\* address of TASKENTRY structure \*/

The TaskNext function fills the specified structure with information about the next task on the task queue.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

|      |                                                                                    |
|------|------------------------------------------------------------------------------------|
| lpte | Points to a <u>TASKENTRY</u> structure to receive information about the next task. |
|------|------------------------------------------------------------------------------------|

Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

Comments

The TaskNext function can be used to continue a walk through the task queue. The walk must have been started by the TaskFirst or TaskFindHandle function.

See Also

TaskFindHandle, TaskFirst, TASKENTRY



### TaskSetCSIP (3.1)

#include toolhelp.h

DWORD TaskSetCSIP(htask, wCS, wIP)

HTASK htask; /\* handle of task \*/

WORD wCS; /\* value in CS register \*/

WORD wIP;

\* value in IP register

/

The TaskSetCSIP function sets the CS:IP value of a sleeping task. When the task is yielded to, it will begin execution at the specified address.

| Parameter | Description                                             |
|-----------|---------------------------------------------------------|
| htask     | Identifies the task to be assigned the new CS:IP value. |
| wCS       | Contains the new value of the CS register.              |
| wIP       | Contains the new value of the IP register.              |

#### Returns

The return value is the previous CS:IP value for the task. The TaskSwitch function uses this value. The return value is NULL if the htask parameter is invalid.

#### Comments

TaskSetCSIP should not be called if htask identifies the current task.

#### See Also

DirectedYield, TaskGetCSIP, TaskSwitch

## TaskSwitch (3.1)

#include toolhelp.h

```
BOOL TaskSwitch(htask, dwNewCSIP)
```

```
HTASK htask;          /* handle of task          */
```

```
DWORD dwNewCSIP;     /* execution address within task */
```

```
execution address within task
```

```
/
```

The TaskSwitch function switches to the given task. The task begins executing at the specified address.

| Parameter | Description                                                                                                                                                   |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| htask     | Identifies the new task.                                                                                                                                      |
| dwNewCSIP | Identifies the address within the given task at which to begin execution. Be very careful that this address is not in a code segment owned by the given task. |

### Returns

The return value is nonzero if the task switch is successful. Otherwise, it is zero.

### Comments

When the task identified by the htask parameter yields, TaskSwitch returns to the calling application.

TaskSwitch changes the CS:IP value of the task's stack frame to the value specified by the dwNewCSIP parameter and then calls the DirectedYield function.

### See Also

DirectedYield, TaskSetCSIP, TaskGetCSIP

## TerminateApp (3.1)

#include toolhelp.h

```
void TerminateApp(htask, wFlags)
```

```
HTASK htask; /* handle of task */
```

```
WORD wFlags; /* termination flags */
```

The TerminateApp function ends the given application instance (task).

| Parameter  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |         |                                                                                               |            |                                                                                                                                                                                                                            |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---------|-----------------------------------------------------------------------------------------------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| htask      | Identifies the task to be ended. If this parameter is NULL, it identifies the current task.                                                                                                                                                                                                                                                                                                                                                                                        |       |         |         |                                                                                               |            |                                                                                                                                                                                                                            |
| wFlags     | Indicates how to end the task. This parameter can be one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |         |                                                                                               |            |                                                                                                                                                                                                                            |
|            | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>UAE_BOX</td><td>Calls the Windows kernel to display the Application Error message box and then ends the task.</td></tr><tr><td>NO_UAE_BOX</td><td>Calls the Windows kernel to end the task but does not display the Application Error message box. The application's interrupt or notification callback function should have displayed an error message, a warning, or both.</td></tr></tbody></table> | Value | Meaning | UAE_BOX | Calls the Windows kernel to display the Application Error message box and then ends the task. | NO_UAE_BOX | Calls the Windows kernel to end the task but does not display the Application Error message box. The application's interrupt or notification callback function should have displayed an error message, a warning, or both. |
| Value      | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |         |                                                                                               |            |                                                                                                                                                                                                                            |
| UAE_BOX    | Calls the Windows kernel to display the Application Error message box and then ends the task.                                                                                                                                                                                                                                                                                                                                                                                      |       |         |         |                                                                                               |            |                                                                                                                                                                                                                            |
| NO_UAE_BOX | Calls the Windows kernel to end the task but does not display the Application Error message box. The application's interrupt or notification callback function should have displayed an error message, a warning, or both.                                                                                                                                                                                                                                                         |       |         |         |                                                                                               |            |                                                                                                                                                                                                                            |

### Returns

This function returns only if htask is not NULL and does not identify the current task.

### Comments

The TerminateApp function unregisters all callback functions registered with the Tool Help functions and then ends the application as if the given task had produced a general-protection (GP) fault or other error.

TerminateApp should be used only by debugging applications, because the function may not free not all objects owned by the ended application.

### See Also

[InterruptRegister](#), [InterruptUnRegister](#), [NotifyRegister](#), [NotifyUnRegister](#)

### TimerCount (3.1)

#include toolhelp.h

BOOL TimerCount(lpti)

TIMERINFO FAR\* lpti; /\* address of structure for execution times \*/

The TimerCount function fills the specified structure with the execution times of the current task and VM (virtual machine).

| Parameter | Description                                                                     |
|-----------|---------------------------------------------------------------------------------|
| lpti      | Points to the <u>TIMERINFO</u> structure that will receive the execution times. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The TimerCount function provides a consistent source of timing information, accurate to the millisecond. In enhanced mode, TimerCount uses the VTD (virtual timer device) to obtain accurate execution times.

In standard mode, TimerCount calls the GetTickCount function, which returns information accurate to one clock tick (approximately 55 ms). TimerCount then reads the hardware timer to estimate how many milliseconds remain until the next clock tick. The resulting time is accurate to 1 ms.

Before calling TimerCount, an application must initialize the TIMERINFO structure and specify its size, in bytes, in the dwSize member.

#### See Also

GetTickCount, TIMERINFO

### Toolhelp Functions (3.1)

|                            |                                                         |
|----------------------------|---------------------------------------------------------|
| <u>ClassFirst</u>          | Retrieves information about first class in class list   |
| <u>ClassNext</u>           | Retrieves information about next class in class list    |
| <u>GlobalEntryHandle</u>   | Retrieves information about given global memory object  |
| <u>GlobalEntryModule</u>   | Retrieves information about specified module segment    |
| <u>GlobalFirst</u>         | Retrieves information about first global memory object  |
| <u>GlobalHandleToSel</u>   | Converts the given global handle to a selector          |
| <u>GlobalInfo function</u> | Retrieves information about the global heap             |
| <u>GlobalNext</u>          | Retrieves information about next global memory object   |
| <u>InterruptRegister</u>   | Installs callback function to handle system interrupts  |
| <u>InterruptUnRegister</u> | Removes function handling system interrupts             |
| <u>LocalFirst</u>          | Retrieves information about first local memory object   |
| <u>LocalInfo function</u>  | Fills structure with information about local heap       |
| <u>LocalNext</u>           | Retrieves information about next local memory object    |
| <u>MemManInfo function</u> | Retrieves information about the memory manager          |
| <u>MemoryRead</u>          | Reads memory from an arbitrary global heap object       |
| <u>MemoryWrite</u>         | Writes memory to an arbitrary global heap object        |
| <u>ModuleFindHandle</u>    | Retrieves information about the given module            |
| <u>ModuleFindName</u>      | Retrieves information about module with specified name  |
| <u>ModuleFirst</u>         | Retrieves information about the first module            |
| <u>ModuleNext</u>          | Retrieves information about the next module             |
| <u>NotifyRegister</u>      | Installs a notification callback function               |
| <u>NotifyUnRegister</u>    | Removes a notification callback function                |
| <u>StackTraceCSIPFirst</u> | Retrieves information about a stack frame               |
| <u>StackTraceFirst</u>     | Retrieves information about the first stack frame       |
| <u>StackTraceNext</u>      | Retrieves information about the next stack frame        |
| <u>SystemHeapInfo</u>      | Retrieves information about the USER and GDI heaps      |
| <u>TaskFindHandle</u>      | Retrieves information about a task                      |
| <u>TaskFirst</u>           | Retrieves information about first task in task queue    |
| <u>TaskGetCSIP</u>         | Returns the next CS:IP value of a sleeping task         |
| <u>TaskNext</u>            | Retrieves information about next task on the task queue |
| <u>TaskSetCSIP</u>         | Sets the CS:IP value of a sleeping task                 |
| <u>TaskSwitch</u>          | Switches to a specific address within a new task        |
| <u>TerminateApp</u>        | Ends the given application instance (task)              |
| <u>TimerCount</u>          | Retrieves execution times of current task and VM        |

## Windows SDK Tools (3.1)

Advanced Debugging (wdeb386.exe)

Analyzing CPU Time: Profiler

Analyzing System Failures: Dr. Watson

Compiling Resources: Resource Compiler

Compressing Files (compress.exe)

Creating WinHelp Databases

Debugging DDE Transactions (ddespy.exe)

Debugging: CodeView for Windows

Expanding Compressed Files (expand.exe)

Module Definition Statements

Monitoring Messages: SPY

Viewing the Heap (heapwalk.exe)

## Analyzing Performance: Profiler

Profiler analyzes applications running with Windows in 386 enhanced mode; however, it cannot analyze applications running with Windows in standard mode.

The following topics describe how to set up and use Profiler:

- [An Overview of Profiler](#)
- [Preparing to Run Profiler](#)
- [Using Profiler Functions](#)
- [Sampling Code](#)
- [Displaying Samples](#)

## Overview of Profiler

Profiler contains the following:

- A sampling utility
- A reporting utility
- A set of functions your application can call

The sampling utility gathers information about the time spent between adjacent labels and records memory addresses of code. The utility is a special device driver, VPROD.386. To run Profiler, install VPROD.386 and then run Windows directly.

Profiler stores the information it gathers in a buffer. It writes the buffer to disk when Windows terminates, producing a CSIPS.DAT file and a SEGENTRY.DAT file in the directory that was your current directory when you started Windows. The CSIPS.DAT file contains statistical samplings of the code segment (CS) and instruction pointer (IP) registers. The SEGENTRY.DAT file contains information about the movement of code segments. Because code segments can be located at different physical addresses during the execution of the program, information from both the CSIPS.DAT and SEGENTRY.DAT files is required to prepare the profiling report.

After the sampling utility has finished gathering information, the SHOWHITS.EXE reporting utility organizes and displays the results.

With Profiler's functions, you start and stop examining code, manage the output of code samples, and get information about Profiler. All applications that Profiler examines must include the two functions that start and stop the sampling of code. Other Profiler functions are optional.



## Preparing to Run Profiler

To profile an application running with Windows in 386 enhanced mode, you can use any system that is capable of running Windows in 386 enhanced mode.

In addition to ensuring that your system is compatible with Profiler, you must do the following:

- 1 Ensure that the Windows directory is defined in your PATH environment variable.
- 2 Include in your application at least the two mandatory Profiler functions ProfStart and ProfStop.  
ProfStart indicates when you want Profiler to start sampling code; ProfStop indicates when you want Profiler to stop sampling. Other Profiler functions are optional.
- 3 Compile your application. Then link the compiled code with the standard Windows libraries, using the appropriate command-line option to prepare a symbol map (.MAP) file that includes PUBLIC symbols. The .MAP file is required by Microsoft Symbol File Generator (MAPSYM). For information about how to create the .MAP file during linking, see the documentation that accompanied your linker. For more information about MAPSYM, see Advanced Debugging in Protected Mode: WDEB386.
- 4 Use MAPSYM to convert the .MAP file to a symbol (.SYM) file.

## Using Profiler Functions

In addition to the mandatory ProfStart and ProfStop functions, Profiler includes functions that determine whether Profiler is installed, specify a rate for sampling, and control the output buffer. Following are the available Profiler functions:

| Function            | Description                                        |
|---------------------|----------------------------------------------------|
| <u>ProfClear</u>    | Discards all buffered Profiler samples.            |
| <u>ProfFinish</u>   | Stops profile sampling and flushes profile buffer. |
| <u>ProfFlush</u>    | Flushes the Profiler sampling buffer to disk.      |
| <u>ProfInsChk</u>   | Determines whether Profiler is installed.          |
| <u>ProfSampRate</u> | Sets the Profiler sampling rate.                   |
| <u>ProfSetup</u>    | Sets Profiler buffer size and sample quantity.     |
| <u>ProfStart</u>    | Starts Profiler sampling.                          |
| <u>ProfStop</u>     | Stops Profiler sampling.                           |

## PROFILER: Sampling Code

To use the Profiler functions, you must first install VPROD.386, a virtual device driver. Your application can call the ProfSetup function to set the size of the output buffer (up to 1064K).

Profiler sampling uses memory that is otherwise available to Windows. Therefore, using Profiler may decrease the performance of the application you are analyzing. By specifying a small output buffer for Profiler, you can reduce the amount of memory used. However, a small output buffer may cause sample loss.

Profiler can write samples to disk only when Windows indicates it is safe to do so. When the sampling buffer is full, Profiler ignores additional samples until the buffer is flushed to disk. To minimize sample loss, either increase the buffer size or periodically call the ProfFlush function.

To profile applications, do the following:

- 1 Install the VPROD.386 driver by adding the following setting to the [386enh] section of your SYSTEM.INI file:

```
device=vprod.386
```

- 2 Run Windows in 386 enhanced mode.
- 3 Run the application you want to profile.
- 4 When you have finished profiling your application, remove the SYSTEM.INI file setting you added in step 1.

## Displaying Samples: SHOWHITS.EXE

To display the data Profiler gathers, run the SHOWHITS.EXE application from the MS-DOS command line. This reporting utility reads CSIPS.DAT, SEGENTRY.DAT, and .SYM files and then organizes and displays the data. The CSIPS.DAT and SEGENTRY.DAT files are located where the sampling utility placed them--that is, in the directory that was your current directory when you started Windows. To ensure that SHOWHITS.EXE can locate these files, either run SHOWHITS.EXE from the same directory or specify full paths for the CSIPS.DAT and SEGENTRY.DAT files. If the .SYM files are not in the current directory, use the /ipath option on the showhits command line to specify the directory or directories containing them.

SHOWHITS.EXE reads .SYM files to match instruction pointer samples with global symbols in the application. When you run SHOWHITS.EXE, the utility searches for .SYM files that contain symbolic names identical to the names of modules that Profiler sampled. Each match is called a hit. If the sampled program is written in the C language, the symbolic names are typically function names. If the sampled program is written in assembly language, the symbolic names can be either procedure names or PUBLIC symbols within procedures.

SHOWHITS.EXE reports the number of times sampling occurred between adjacent symbols.

The syntax for the showhits command line is as follows:

```
showhits [/ipath [/ipath [...]]] [cs_file] [seg_file]
```

Following are the command-line options and parameters:

/ipath Specifies one or more directories to search for .SYM files. SHOWHITS.EXE loads all .SYM files from the specified directories, regardless of their relevance to the application you are profiling. The default value is the current directory.

cs\_file Specifies the full path of the CSIPS.DAT file. If no path is specified, SHOWHITS.EXE looks for the file in the current directory.

seg\_file Specifies the full path of the SEGENTRY.DAT file. If no path is specified, SHOWHITS.EXE looks for the file in the current directory.

SHOWHITS.EXE displays information about hits, which are instruction pointer samples, in the following four categories:

| Category              | Description                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Unrecognized segments | A list of instruction pointer values that occur within segments for which there are no symbols of module names. Unrecognized segments are typically code for device drivers, terminate-and-stay-resident (TSR) programs, and other code that Windows does not use.                                                                                                                                                        |
| Known segments        | The number of hits that occur within known modules. Hits on known segments typically include counts for the application and counts for such Windows modules as KERNEL, GDI, and DISPLAY. Profiler also counts hits in MS-DOS and the read-only memory (ROM) basic input-and-output system (BIOS). In addition to displaying hits, SHOWHITS.EXE lists the total number of hits and the segment's percentage of total hits. |
| Breakdown             | A detailed breakdown of the hits between labels of the modules for which SHOWHITS.EXE finds .SYM files. SHOWHITS.EXE also displays the total number of hits and the percentage of the total number.                                                                                                                                                                                                                       |
| Summary               | A list of the top hits.                                                                                                                                                                                                                                                                                                                                                                                                   |

The following example illustrates a profiling-report display:

```
Here are the Hits for Unrecognized Segments
```

```
Here are the Hits for Known Segments
```

```
0.3% 3 Hits on SYSTEM!  
0.5% 5 Hits on HELLO!  
76.5% 786 Hits on DISPLAY!  
11.3% 116 Hits on GDI!  
11.5% 118 Hits on KERNEL!
```

```
1028 TOTAL HITS
```

```
HELLO!_TEXT
```

0.4% 4 Hits between labels \_HelloPaint and \_HelloInit  
0.1% 1 Hits between labels \_\_cintDIV and \_\_fptrap

Profiler Summary (Top 10 Hits):

0.4% 4 HELLO! \_TEXT! \_HelloPaint - \_HelloInit  
0.1% 1 HELLO! \_TEXT! \_\_cintDIV - \_\_fptrap

## Advanced Debugging in Protected Mode: WDEB386

Microsoft Windows 80386 Debugger (WDEB386.EXE) is used to test and debug Windows applications and dynamic-link libraries (DLLs) running with the Microsoft Windows operating system in standard or 386 enhanced mode. With 80386 Debugger commands, you can inspect and manipulate test code and environment status, install breakpoints, and perform other debugging operations.

Although 80386 Debugger offers debugging features not available in CodeView for Windows (CVW), 80386 Debugger lacks the convenient window interface of CVW and does not provide source-level debugging.

To use 80386 Debugger, you must have a serial terminal connected to the computer on which you are running the debugger and test application.

The following topics describe how to set up and use WDEB386:

Preparing Symbol Files for the 80386 Debugger

Starting the Debugger

Entering the Debugger

Debugger Command Format

Common Command Directory

## Preparing Symbol Files for the 80386 Debugger

### Preparing Symbol Files for 80386 Debugger

To prepare application symbol files, perform the following steps:

- 1 Compile your C-language source files, using the appropriate command-line option to generate object files with line-number information for use by 80386 Debugger. For more information about compiler options, see the documentation that accompanied your compiler.
- 2 Link the compiled code with the standard Windows libraries, using the appropriate command-line option to prepare a symbol map (.MAP) file that includes PUBLIC symbols. The map file is required by Microsoft Symbol File Generator (MAPSYM).  
You may also want to use the linker option for display of line-number information. For more information about linker options, see the documentation that accompanied your linker.
- 3 Run MAPSYM to create a symbol file for symbolic debugging. MAPSYM converts the contents of your application's symbol map (.MAP) file into a form suitable for loading with 80386 Debugger; then MAPSYM copies the result to a symbol (.SYM) file.

Following is the command-line syntax for MAPSYM:

```
mapsym [/l]/[n] mapfilename
/l          Directs MAPSYM to display information on the screen about the conversion. The
           information includes the names of groups defined in the application, the
           application start address, the number of segments, and the number of symbols per
           segment.
/n          Directs MAPSYM to ignore line-number information in the map file. The
           resulting symbol file contains no line-number information.
mapfilename Specifies the filename for a symbol map file that was created during linking. If
           you do not give a filename extension, .MAP is assumed. If you do not give a full
           path, the current directory and drive are assumed. MAPSYM creates a new
           symbol file having the same name as the map file but with the .SYM extension.
```

In the following example, MAPSYM uses the symbol information in FILE.MAP to create FILE.SYM in the current directory on the current drive:

```
mapsym /l file.map
```

Information about the conversion is sent to the screen.

Note: MAPSYM always places the new symbol file in the current directory on the current drive. MAPSYM can process up to 10,000 symbols for each segment in the application and up to 1024 segments.

## Starting the Debugger

### Starting 80386 Debugger

A three-wire null modem cable is the minimum cable requirement for the serial terminal. In a three-wire null modem cable, the TxD (transmit data) and RxD (receive data) lines are in opposite positions at the two ends of the cable, but the signal ground is connected straight through.

The command-line syntax is as follows:

```
wdeb386 [/C:comport] [/D:"commands"] [/F:filename] [/N] [/T:hhhh] [/S:symfile] [/V[P]] [X] winfile  
[parameters]
```

Following are the command-line options and parameters:

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /C:comport    | Specifies a COM port for debugger output. If this option is not specified, 80386 Debugger checks first for COM2. If COM2 is not found, the debugger then checks for COM1. If neither COM1 nor COM2 exists, the debugger checks for any other COM port in the read-only memory (ROM) data area (40:0). A three-wire null modem cable is all that is needed for terminal connection; no DTR (data-terminal-ready) and CTS (clear-to-send) handshaking is used.                                                                                                                                                                                                                                                                                    |
| /D:"commands" | Carries out the 80386 Debugger command line specified by the string enclosed in quotation marks. Spaces, semicolons (;), and other punctuation can be included in the command string. To use a single quote (') on the command line, use double quotation marks (") before and after the single quotation mark.<br><br>The commands specified in this option are carried out after symbols are loaded. This means you can set breakpoints in code even before the code has been loaded. Before a segment or module has been loaded or defined, breakpoints can be set on the logical address (a combination of map number and group number) until the segment or module is defined, at which point the breakpoint turns into a real breakpoint. |
| /F: filename  | Specifies a file containing command-line options for 80386 Debugger. Maximum file size is 4K, and the input file cannot contain the /F option.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| /N            | Sets the following options:<br><br>dislwr<br>codebytes<br>symaddr<br>int3line<br>newvec<br>newreg<br>newprompt                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| /S: symfile   | Specifies a symbol file to be loaded. This option can be repeated to load more than one symbol file. If the symbol files are not in your current directory, you must supply a full path, because 80386 Debugger does not use the PATH environment variable to locate any of the files supplied on the command line.<br><br>When memory is low, you can use more symbol files by running 80386 Debugger in the Windows directory and specifying the full path of WIN386.EXE (such as \WINDOWS\SYSTEM\WIN386.EXE) instead of WIN.COM.                                                                                                                                                                                                             |
| /T:hhhh       | Sets the port number for the timing card. (The default number is 250h.)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| /V            | Enables verbose mode, which displays messages indicating which segments are being being loaded. This option displays the messages for both Windows in 386 enhanced mode and Windows applications.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| /VP           | Enables verbose mode, which displays messages indicating which segments are being loaded. This option displays the messages for applications only.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| /X            | Causes symbols to be loaded into Extended Memory Specification (XMS) memory. This option has no effect with Windows version 3.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| winfile       | Specifies the Windows application to run under 80386 Debugger control. You will usually specify WIN.COM.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| parameters    | Specifies any parameters to be passed to the application.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

Note: The length of the command line cannot exceed 128 characters.

Following are two examples of valid commands:



```
wdeb386 /V /S:\windows\system\krnl286.sym /S:myapp.sym  
\windows\win.com /s myapp
```

```
wdeb386 /C:1 /S:krnl386.sym /s:user.sym /S:\myapp\myapp.sym  
\windows\win.com /3 myapp
```

You can start 80386 Debugger as a device driver by placing the following line in your CONFIG.SYS file:

```
device=c:\windev\wdeb386.exe
```

You must specify the full path to the WDEB386.EXE file. You can specify any command-line options on the line with device= (for example, you can load symbol files).

## Entering the Debugger

### Entering 80386 Debugger

To enter 80386 Debugger at any time interrupts are not disabled, press the CTRL+C key combination on the debugging terminal. A nonmaskable interrupt (NMI) can be used to enter the debugger even when interrupts are disabled.

An int 3 instruction or a call to the Windows DebugBreak function passes control to the 80386 Debugger.

When a Windows application running in standard or 386 enhanced mode attempts to read or write memory with a bad selector, beyond a selector limit, or with a selector set to 0, a general protection (GP) fault occurs.

In such cases, Windows displays a dialog box notifying the user of a problem. When 80386 Debugger is loaded, the dialog box has a Cancel button. If the user chooses the Cancel button, Windows passes control to the debugger at the instruction that caused the fault with a display of the following form:

```
GENERAL PROTECTION VIOLATION
AX=00000000 BX=00002136 CX=06040079 DX=00001EF5 SI=000000C3 DI=
00002283
IP=00000028 SP=80012126 BP=0000212C CR2=80501FFC CR3=0293 IOPL=0 F--- -
-
CS=0915 SS=091D DS=091D ES=0000 FS=0000 GS=0000 -- NV UP EI PLZR NA PE
NC
00AD:00000FA0 MOV BX, WORD PTR ES:[BX]
ES:65DF=INV:0003#
```

For more information about commands shown in the remaining examples in this section, see Command Parameters.

You can determine the cause of the GP fault by looking at the value and the limit of the selector. To dump the local descriptor table (LDT) entry, you can use a command of the following form:

```
dl selector
```

The ability to continue execution depends on the cause of the fault. If the fault was caused by reading or writing beyond the selector limit, it may be possible to skip the instruction by incrementing the IP register.

To determine how many bytes the instruction contains, you may need to display the actual code bytes when disassembling the instruction. To do this, use the following commands:

```
y codebytes
r
```

If the fault is caused by a critical logic error, such as trying to use a selector for a temporary variable, there probably is no way to continue execution of the application. You may need to restart the computer.

## Debugger Command Format

### Command Syntax

To enter 80386 Debugger commands, you use a debugging terminal rather than your computer's keyboard.

Commands and parameters are not case-sensitive.

If a syntax error occurs in a debugger command, 80386 Debugger redisplay the command line and indicates the error with a caret (^) and the word Error, as in the following example:

```
A100
 ^ Error
```

### Command Keys

Following are the command keys:

| Key    | Action                                                           |
|--------|------------------------------------------------------------------|
| CTRL+A | Repeats the previous command.                                    |
| CTRL+C | Halts 80386 Debugger output, and returns to the debugger prompt. |
| CTRL+S | Freezes an 80386 Debugger display.                               |
| CTRL+Q | Restarts the display.                                            |

If the target system is executing code, CTRL+S and CTRL+Q are ignored.

### Command Parameters

You can separate 80386 Debugger command parameters with delimiters (spaces or commas), but a delimiter is required only between two consecutive hexadecimal values. The following commands are equivalent:

```
dCS:100 110
d CS:100 110
d ,CS:100 ,110
```

Following are the parameters you can use with 80386 Debugger commands:

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| addr       | Represents an address parameter in one of four forms. For more information about the operators shown in the following address forms, see Section 5.4.3, "Binary and Unary Operators."<br>Address            Mode<br>#1f:02C0           Protected-mode address (selector:offset)<br>%31020             Linear address<br>%%31020            Physical address<br>&0100:02FF         Real-mode address (segment:offset)<br>Any of these specified address forms overrides the current address type. |
| byte       | Specifies a two-digit hexadecimal value.                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| cmds       | Specifies an optional set of debugger commands to be executed with the bp (Set Breakpoint) or j (Conditionally Execute) command.                                                                                                                                                                                                                                                                                                                                                                 |
| count      | Specifies a count. Valid values depend on the command with which this parameter is being used.                                                                                                                                                                                                                                                                                                                                                                                                   |
| dword      | Represents an eight-digit (4-byte) hexadecimal value. The <u>DWORD</u> data type is most commonly used as a physical address.                                                                                                                                                                                                                                                                                                                                                                    |
| expr       | Represents a combination of parameters and operators that evaluates to an 8-bit, 16-bit, or 32-bit value. An expr parameter can be used as a value in any command. An expr parameter can combine any symbol, number, or address with any of the binary and unary operators.                                                                                                                                                                                                                      |
| flags      | Specifies one or more conditions. Valid conditions depend on the command with which this parameter is being used.                                                                                                                                                                                                                                                                                                                                                                                |
| group-name | Specifies the name of a group that contains the map symbols you want to display.                                                                                                                                                                                                                                                                                                                                                                                                                 |
| list       | Specifies a series of byte values or a string. The list parameter must be the last parameter on the command line. Following is an example of the f (Fill) command with a list parameter:                                                                                                                                                                                                                                                                                                         |

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            | fCS:100 42 45 52 54 41                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| map-name   | Specifies the name of a symbol map file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| name-chars | Specifies one or more characters.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| number     | Specifies a numeric value. Valid values depend on the command with which this parameter is being used.                                                                                                                                                                                                                                                                                                                                                                                                                         |
| object     | Specifies a handle, a selector, or (in 386 enhanced mode) a heap address.                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| option     | Specifies an option. Valid options depend on the command with which this parameter is being used.                                                                                                                                                                                                                                                                                                                                                                                                                              |
| range      | Specifies the block of memory on which the command should operate. The range parameter can be two addresses (addr addr); or it can be one address and a length (addr L word, where word is the number of items on which the command should operate; 80h is the default value). Following are three valid examples:<br><br>CS:100 110<br>CS:100 L 10<br>CS:100<br>The limit for range is 10000h. To specify a word of 10000h using only four digits, use 0000h or 0h.                                                           |
| reg        | Specifies the name of a microprocessor register.                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| string     | Represents any number of characters enclosed in single quotation marks (') or double quotation marks ("). For quotation marks that must appear within string, you must use two sets of quotation marks. For example, the following strings are valid:<br><br>'This 'string' is OK.'<br>\"This \"string\" is OK.\"<br>However, the following strings are not valid:<br><br>\"This \"string\" is not OK.\"<br>'This 'string' is not OK.\"<br>The ASCII values of the characters in the string are used as a list of byte values. |
| word       | Specifies a four-digit (2-byte) hexadecimal value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

### Stopping Execution

The BreakInDebugVxD entry in the [386Enh] section of SYSTEM.INI controls where WDEB386 stops execution when CTRL+ALT+SYSREQ is pressed. The default setting for this entry is FALSE, which causes WDEB386 to stop in application code. When the setting is TRUE, WDEB386 stops at the current instruction, which is frequently in WIN386.EXE or in a VxD.

### Binary and Unary Operators

Following, in descending order of precedence, are the binary operators that can be used in 80386 Debugger commands:

| Operator | Meaning                                   |
|----------|-------------------------------------------|
| ()       | Parentheses                               |
| :        | Address binder                            |
| *        | Multiplication                            |
| /        | Integer division                          |
| MOD      | Modulus (remainder)                       |
| +        | Addition                                  |
| -        | Subtraction                               |
| >        | Greater-than relational operator          |
| <        | Less-than relational operator             |
| >=       | Greater-than/equal-to relational operator |
| <=       | Less-than/equal-to relational operator    |
| ==       | Equal-to relational operator              |
| !=       | Not-equal-to relational operator          |
| AND      | Bitwise Boolean AND                       |

|     |                              |
|-----|------------------------------|
| XOR | Bitwise Boolean exclusive OR |
| OR  | Bitwise Boolean OR           |
| &&  | Logical AND                  |
|     | Logical OR                   |

Following, in descending order of precedence, are the unary operators that can be used in 80386 Debugger commands:

| Operator | Meaning                                                                             |
|----------|-------------------------------------------------------------------------------------|
| &(seg)   | Address of segment value                                                            |
| #(sel)   | Address of selector value                                                           |
| %%(phy)  | Address as a physical value                                                         |
| %(lin)   | Address as a linear value                                                           |
| -        | Two's complement                                                                    |
| !        | Logical NOT operator                                                                |
| NOT      | One's complement                                                                    |
| SEG      | Segment address of operand                                                          |
| OFF      | Address offset of operand                                                           |
| BY       | Low-order byte from given address                                                   |
| WO       | Low-order word from given address                                                   |
| DW       | Doubleword from given address                                                       |
| POI      | Pointer (4 bytes) from given address--this operator works only with 16:16 addresses |
| PORT     | 1 byte from given port                                                              |
| WPORT    | Word from given port                                                                |

### Regular Expressions

The set of regular expressions that 80386 Debugger supports for matching symbols is similar to the set supported by UNIX grep. The 80386 Debugger set includes a few enhancements.

Following are the 80386 Debugger wildcards:

| Wildcard | Description                                                      |
|----------|------------------------------------------------------------------|
| .        | Matches any single character.                                    |
| []       | Defines a character class; matches a set or range of characters. |
| ^        | Negates a character class.                                       |

Following are the 80386 Debugger postfix operators:

| Operator | Description                                                                        |
|----------|------------------------------------------------------------------------------------|
| *        | Causes the previous wildcard or single character to match zero or more characters. |
| #        | Matches zero or one.                                                               |
| +        | Plus sign, matches one or more.                                                    |

Anywhere a symbol is accepted, a regular expression can be used. If there is more than one match, a list of matching symbols is displayed and you must select the proper symbol. The symbol match is not case-sensitive.

The asterisk (\*), number sign (#), and plus sign (+) are already math expression operators. To be recognized as a regular expression operator, each of these characters must be immediately preceded by an escape character--the backslash (\). The period (.), opening bracket ([), and closing bracket (]) do not require escape characters. Anything inside the brackets of a character class does not have to be escaped.

Following are valid character classes:

```
[ a-z ]
[ ; * + # ]
```

Characters are escaped at two levels: in the expression evaluator and in the regular expression parser. A character special to the expression evaluator (\*, #, +, or \) must be escaped to make it to the regular expression parser. If a character special to the regular expression parser must be escaped (for example, to match symbols with \* or # in them), it must be escaped twice. If a backslash is needed in an expression, it must be double escaped.

Following are sample regular expressions:

| Regular expression    | Description                                                                     |
|-----------------------|---------------------------------------------------------------------------------|
| <code>sym.*</code>    | Matches any symbols beginning with the string <code>sym</code> .                |
| <code>sym\*</code>    | Matches <code>sym</code> alone and <code>sym</code> followed by any characters. |
| <code>.*sym.*</code>  | Matches any symbols containing the string <code>sym</code> .                    |
| <code>sym[0-9]</code> | Matches <code>sym0</code> , <code>sym1</code> , <code>sym2</code> , and so on.  |
| <code>sym\ \*</code>  | Matches <code>sym*</code> .                                                     |
| <code>sym\\ </code>   | Matches <code>sym\</code> .                                                     |
| <code>sym\\ .*</code> | Matches any symbols beginning with the string <code>sym\</code> .               |

## Common Command Directory

### Common Commands

This section documents the commands available in all environments in which you can use 80386 Debugger. A command that begins with a period (.) is called a dot command.

| Command   | Description                                                      |
|-----------|------------------------------------------------------------------|
| <u>?</u>  | Display expression, or display help menu.                        |
| .?        | Display external commands.                                       |
| .b        | Set baud rate for COM port.                                      |
| .df       | Display global free list.                                        |
| .dg       | Display global heap.                                             |
| .dh       | Display local heap.                                              |
| .dm       | Display global module list.                                      |
| .dq       | Dump task queues.                                                |
| .du       | Display list of least recently used (LRU) global memory objects. |
| .reboot   | Restart target system.                                           |
| <b>bc</b> | Clear breakpoint.                                                |
| <b>bd</b> | Disable breakpoint.                                              |
| <b>be</b> | Enable breakpoint.                                               |
| <b>bl</b> | List breakpoints.                                                |
| <b>bp</b> | Set breakpoint.                                                  |
| <b>br</b> | Set breakpoint on debug register.                                |
| <b>c</b>  | Compare memory locations.                                        |
| <b>d</b>  | Display memory.                                                  |
| <b>db</b> | Display bytes.                                                   |
| <b>dd</b> | Display doublewords.                                             |
| <b>dg</b> | Display global descriptor table (GDT).                           |
| <b>di</b> | Display interrupt descriptor table (IDT).                        |
| <b>dl</b> | Display local descriptor table (LDT).                            |
| <b>dp</b> | Display page directory and page tables.                          |
| <b>dt</b> | Display task state segment (TSS).                                |
| <b>dw</b> | Display words.                                                   |
| <b>e</b>  | Enter byte.                                                      |
| <b>f</b>  | Fill memory.                                                     |
| <b>g</b>  | Go.                                                              |
| <b>h</b>  | Perform hexadecimal arithmetic.                                  |
| <b>i</b>  | Display 1 byte of input.                                         |
| <b>j</b>  | Conditionally execute command.                                   |
| <b>k</b>  | Display current stack frame.                                     |
| <b>ka</b> | Set backtrace argument.                                          |
| <b>kt</b> | Display stack frame of task.                                     |
| <b>la</b> | List absolute symbols.                                           |
| <b>lg</b> | List groups.                                                     |
| <b>lm</b> | List maps.                                                       |
| <b>ln</b> | List nearest symbol.                                             |
| <b>ls</b> | List symbols.                                                    |
| <b>m</b>  | Move memory.                                                     |
| <b>o</b>  | Write output to a port.                                          |
| <b>p</b>  | Execute instruction, returning from any call or interrupt.       |
| <b>r</b>  | Display register.                                                |
| <b>s</b>  | Search for a byte.                                               |
| <b>t</b>  | Execute instruction.                                             |

|           |                                                      |
|-----------|------------------------------------------------------|
| <u>u</u>  | Disassemble bytes.                                   |
| <u>v</u>  | Display debugger version.                            |
| <u>vc</u> | Clear interrupt vector.                              |
| <u>vl</u> | List debugger interrupt vectors.                     |
| <u>vo</u> | List debugger interrupt vectors in specified format. |
| <u>vs</u> | Add debugger interrupt vector (not at ring 0).       |
| <u>vt</u> | Add debugger interrupt vector.                       |
| <u>w</u>  | Change active map list.                              |
| <u>wa</u> | Add map to active list.                              |
| <u>wr</u> | Remove map from active list.                         |
| <u>y</u>  | Change debugger configuration.                       |
| <u>z</u>  | Zap embedded int 1 or int 3 instruction.             |
| <u>zd</u> | Execute default command string.                      |
| <u>zl</u> | Display default command string.                      |
| <u>zs</u> | Change default command string.                       |



## ? WDEB386 command

? [[option.]expr]&? ["string", expr, expr, [...]]

The ? command evaluates an expression and displays the result.

The ? command with no arguments displays a list of commands and syntax recognized by the debugger.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                             |           |        |   |             |   |         |   |         |   |       |   |       |   |        |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|--------|---|-------------|---|---------|---|---------|---|-------|---|-------|---|--------|
| option    | Specifies the format in which to display the expression specified by expr. The option parameter can be one of the following characters:                                                                                                                                                                                                                                                                 |           |        |   |             |   |         |   |         |   |       |   |       |   |        |
|           | <table border="0"> <thead> <tr> <th>Character</th> <th>Format</th> </tr> </thead> <tbody> <tr> <td>h</td> <td>Hexadecimal</td> </tr> <tr> <td>d</td> <td>Decimal</td> </tr> <tr> <td>t</td> <td>Decimal</td> </tr> <tr> <td>o</td> <td>Octal</td> </tr> <tr> <td>q</td> <td>Octal</td> </tr> <tr> <td>y</td> <td>Binary</td> </tr> </tbody> </table>                                                    | Character | Format | h | Hexadecimal | d | Decimal | t | Decimal | o | Octal | q | Octal | y | Binary |
| Character | Format                                                                                                                                                                                                                                                                                                                                                                                                  |           |        |   |             |   |         |   |         |   |       |   |       |   |        |
| h         | Hexadecimal                                                                                                                                                                                                                                                                                                                                                                                             |           |        |   |             |   |         |   |         |   |       |   |       |   |        |
| d         | Decimal                                                                                                                                                                                                                                                                                                                                                                                                 |           |        |   |             |   |         |   |         |   |       |   |       |   |        |
| t         | Decimal                                                                                                                                                                                                                                                                                                                                                                                                 |           |        |   |             |   |         |   |         |   |       |   |       |   |        |
| o         | Octal                                                                                                                                                                                                                                                                                                                                                                                                   |           |        |   |             |   |         |   |         |   |       |   |       |   |        |
| q         | Octal                                                                                                                                                                                                                                                                                                                                                                                                   |           |        |   |             |   |         |   |         |   |       |   |       |   |        |
| y         | Binary                                                                                                                                                                                                                                                                                                                                                                                                  |           |        |   |             |   |         |   |         |   |       |   |       |   |        |
|           | If option is given, a period (.) must be used to separate option and expr. If option is not given, the command displays all formats, an ASCII character representation, and whether the expression is TRUE or FALSE.                                                                                                                                                                                    |           |        |   |             |   |         |   |         |   |       |   |       |   |        |
| expr      | Specifies an expression consisting of one or more addresses, numbers, and operators. The operators in the expression can be any of the 80386 Debugger operators listed in Section 5.4.3, "Binary and Unary Operators." The addresses in the expression can be 32-bit physical addresses or protected-mode addresses (selector:offset). The number sign (#) operator overrides the current address type. |           |        |   |             |   |         |   |         |   |       |   |       |   |        |

string Specifies a printf formatting string. Supported printf format characters are as follows:

| Format character                           | Meaning          |
|--------------------------------------------|------------------|
| %%                                         | %                |
| %c                                         | Character        |
| %[-][+][ ][0][width][.precision][p][n]d    | Decimal          |
| %[-][0][width][.precision][p][n]u          | Unsigned decimal |
| %[-][#][0][width][.precision][p][n]x       | Hexadecimal      |
| %[-][#][0][width][.precision][p][n]X       | Hexadecimal      |
| %[-][0][width][.precision][p][n]o          | Octal            |
| %[-][0][width][.precision][p][n]b          | Binary           |
| %[-][width][.precision][a]s                | String           |
| %[-][width][.precision][a][p][n][L][H][N]S | Symbol           |
| %[-][width][.precision][a][p][n][L][H][N]G | Group:symbol     |
| %[-][width][.precision][a][p][n][L][H][N]M | Map:group:symbol |
| %[-][width][.precision][a][p][n][L][H][N]A | Address          |

Specifying an asterisk (\*) for the width or precision parameter causes the field width or precision, respectively, to be picked up from the next parameter. Decimal values can also be specified for the width and precision parameters.

The following escape sequences are supported:

| Escape sequence | Description            |
|-----------------|------------------------|
| \a              | Alert (bell) character |
| \b              | Backspace              |
| \n              | New line               |
| \r              | Carriage return        |
| \t              | Horizontal tab         |

The following table describes the optional prefixes:

| Prefix | Format character(s) | Meaning               |
|--------|---------------------|-----------------------|
| a      | s,S,G,M,A           | Address argument size |
| H      | S,G,M,A             | 16-bit offset         |
| L      | S,G,M,A             | 32-bit offset         |

|   |             |                                 |
|---|-------------|---------------------------------|
| N | S,G,M,A     | Offset only                     |
| p | S,G,M       | Get the previous symbol         |
| n | S,G,M       | Get the next symbol             |
| p | A           | Get the previous symbol address |
| n | A           | Get the next symbol address     |
| p | d,u,x,X,o,b | Get the previous symbol offset  |
| n | d,u,x,X,o,b | Get the next symbol offset      |

#### Example

The following example looks up the physical address of selector 1Fh in the current local descriptor table (LDT) and adds 220h to it:

```
?% (#001F:0220)
```

The following example displays the value of the expression DS:SI + BX:

```
? ds:si+bx
```

The debugger returns a display similar to the following:

```
987A:000001B3 %00098953 %%00098953
```

The following example displays the value of the arithmetic expression 3\*4:

```
? 3*4
```

The debugger returns the following display:

```
0Ch 12T 14Q 00001100Y '.' TRUE
```

.? WDEB386 command

.?

The .? command displays a list of external commands. These commands are part of 80386 Debugger, but they are specific to the environment in which the debugger is running.

## .b WDEB386 command

.b number [addr]

The .b command sets the baud rate for the debugging port (COM2).

| Parameter | Description                                                                                                                                                                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| number    | Specifies the baud rate. It can be one of the following values: 150, 300, 600, 1200, 2400, 4800, 9600, or 19200. Because the default radix for the debugger is 16, you must type t after the number to indicate a decimal value.              |
| addr      | Specifies 1 for COM1 or 2 for COM2; anything else is taken as a base port address. If there is no COM2, 80386 Debugger checks for COM1 and then for any other COM port address in the read-only memory (ROM) data area to use as the console. |

### Example

The following example sets the baud rate to 1200:

```
#.b 1200t
```

## .df WDEB386 command

.df

The .df command displays a list of the free global memory objects in the global heap.

The list has the following form:

address: size owner [chain]

address      Specifies the selector of the memory in standard mode. In 386 enhanced mode, the address field specifies physical and heap addresses.

size         Specifies the size, in paragraphs (multiples of 16 bytes), of the object in standard mode. In 386 enhanced mode, the size field specifies the size of the object, in bytes.

owner        Always specifies that the module is free.

chain        Specifies the previous and next addresses in the list of least recently used (LRU) objects. 80386 Debugger displays the addresses only if the segment is movable and discardable.

## .dg WDEB386 command

.dg [object]

The .dg command displays a list of the global memory objects in the global heap.

| Parameter | Description                                                                                                                          |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------|
| object    | Specifies the first object to be listed. The object parameter can be a handle, a selector, or (in 386 enhanced mode) a heap address. |

The list has the following form:

address: size segment-type owner [handle flags chain]

| address      | Specifies the selector of the memory in standard mode. In 386 enhanced mode, the address field specifies physical and heap addresses.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|---------|------|-----------------------------------------|------|---------------------------------------------------------------------------------------------------|------|----------------------------------------------------------------------------------------|------|--------------------------------|----------|--------------------------------------------------------|
| size         | Specifies the size, in paragraphs (multiples of 16 bytes), of the object in standard mode. In 386 enhanced mode, the size field specifies the size of the object, in bytes.                                                                                                                                                                                                                                                                                                                                                                                                                                             |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |
| segment-type | Specifies the type of object. The type can be any one of the following: <table><thead><tr><th>Segment type</th><th>Meaning</th></tr></thead><tbody><tr><td>CODE</td><td>Segment contains application code.</td></tr><tr><td>DATA</td><td>Segment contains application data and possible stack and local heap data.</td></tr><tr><td>FREE</td><td>Segment belongs to pool of free memory objects ready for allocation by an application.</td></tr><tr><td>PRIV</td><td>Segment contains private data.</td></tr><tr><td>SENTINAL</td><td>Segment marks the beginning or end of the global heap.</td></tr></tbody></table> | Segment type | Meaning | CODE | Segment contains application code.      | DATA | Segment contains application data and possible stack and local heap data.                         | FREE | Segment belongs to pool of free memory objects ready for allocation by an application. | PRIV | Segment contains private data. | SENTINAL | Segment marks the beginning or end of the global heap. |
| Segment type | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |
| CODE         | Segment contains application code.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |
| DATA         | Segment contains application data and possible stack and local heap data.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |
| FREE         | Segment belongs to pool of free memory objects ready for allocation by an application.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |
| PRIV         | Segment contains private data.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |
| SENTINAL     | Segment marks the beginning or end of the global heap.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |
| owner        | Specifies the module name of the application or library that allocated the memory object. The acronym PDB is used for memory objects that represent process descriptor blocks. These blocks contain execution information about applications.                                                                                                                                                                                                                                                                                                                                                                           |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |
| handle       | Specifies the handle of the global memory object. If 80386 Debugger displays no handle, the segment is fixed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |
| flags        | Specifies either of the following: <table><thead><tr><th>Flag</th><th>Meaning</th></tr></thead><tbody><tr><td>D</td><td>The segment is movable and discardable.</td></tr><tr><td>L</td><td>The segment is locked. If the segment is locked, the lock count appears to the right of the flag.</td></tr></tbody></table> <p>If 80386 Debugger displays a handle but no flag, the segment is movable but not discardable.</p>                                                                                                                                                                                              | Flag         | Meaning | D    | The segment is movable and discardable. | L    | The segment is locked. If the segment is locked, the lock count appears to the right of the flag. |      |                                                                                        |      |                                |          |                                                        |
| Flag         | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |
| D            | The segment is movable and discardable.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |
| L            | The segment is locked. If the segment is locked, the lock count appears to the right of the flag.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |
| chain        | Specifies the previous and next addresses in the list of least recently used (LRU) objects. Addresses are displayed only if the segment is movable and discardable (specified by the D flag).                                                                                                                                                                                                                                                                                                                                                                                                                           |              |         |      |                                         |      |                                                                                                   |      |                                                                                        |      |                                |          |                                                        |

## .dh WDEB386 command

.dh

The .dh command displays a list of the local memory objects in the local heap (if any) belonging to the current data segment. The command uses the current value of the DS register to locate the data segment and check for a local heap.

The list of memory objects has the following form:

offset: size { BUSY | FREE }

offset      Specifies the address offset from the beginning of the data segment to the local memory object.

size        Specifies the size of the object, in bytes.

If BUSY is displayed, the object has been allocated and is currently in use. If FREE is displayed, the object is in the pool of free objects ready to be allocated by the application. A special memory object, SENTINAL, may also be displayed.

## .dm WDEB386 command

.dm

The .dm command displays a list of the global modules in the global heap.

The list has the following form:

module-handle module-type module-name filename

module-handle        Specifies the handle of the module.

module-type         Specifies either a dynamic-link library (DLL) or the name of the application you are debugging.

module-name         Specifies the name of the module.

filename             Specifies the name of the file from which you loaded the application.



## .dq WDEB386 command

.dq

The .dq command displays a list containing information about the various task queues supported by the system.

The list has the following form:

task-descriptor-block stack-segment:stack-pointer number-of-events

priority internal-messaging-information module

|                                |                                                                                                                             |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| task-descriptor-block          | Specifies the selector or segment address.<br>The task descriptor block is identical to the process descriptor block (PDB). |
| stack-segment:stack-pointer    | Specifies the stack segment and pointer.                                                                                    |
| number-of-events               | Specifies the number of events waiting for the segment.                                                                     |
| priority                       | Specifies the priority of the segment.                                                                                      |
| internal-messaging-information | Specifies information about internal messages.                                                                              |
| module                         | Specifies the module name.                                                                                                  |

## .du WDEB386 command

.du

The .du command displays a list of the least recently used (LRU) global memory objects in the global heap.

The list has the following form:

address: size segment-type owner [handle flags chain]

| address      | Specifies the selector of the memory in standard mode. In 386 enhanced mode, the address field specifies physical and heap addresses.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |              |         |      |                                    |      |                                                                           |      |                                                                                        |      |                                |          |                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|---------|------|------------------------------------|------|---------------------------------------------------------------------------|------|----------------------------------------------------------------------------------------|------|--------------------------------|----------|--------------------------------------------------------|
| size         | Specifies the size, in paragraphs (multiples of 16 bytes), of the object in standard mode. In 386 enhanced mode, the size field specifies the size of the object, in bytes.                                                                                                                                                                                                                                                                                                                                                                                                                                                |              |         |      |                                    |      |                                                                           |      |                                                                                        |      |                                |          |                                                        |
| segment-type | Specifies the type of object. The type can be any one of the following:<br><table><thead><tr><th>Segment type</th><th>Meaning</th></tr></thead><tbody><tr><td>CODE</td><td>Segment contains application code.</td></tr><tr><td>DATA</td><td>Segment contains application data and possible stack and local heap data.</td></tr><tr><td>FREE</td><td>Segment belongs to pool of free memory objects ready for allocation by an application.</td></tr><tr><td>PRIV</td><td>Segment contains private data.</td></tr><tr><td>SENTINAL</td><td>Segment marks the beginning or end of the global heap.</td></tr></tbody></table> | Segment type | Meaning | CODE | Segment contains application code. | DATA | Segment contains application data and possible stack and local heap data. | FREE | Segment belongs to pool of free memory objects ready for allocation by an application. | PRIV | Segment contains private data. | SENTINAL | Segment marks the beginning or end of the global heap. |
| Segment type | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |              |         |      |                                    |      |                                                                           |      |                                                                                        |      |                                |          |                                                        |
| CODE         | Segment contains application code.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |              |         |      |                                    |      |                                                                           |      |                                                                                        |      |                                |          |                                                        |
| DATA         | Segment contains application data and possible stack and local heap data.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |              |         |      |                                    |      |                                                                           |      |                                                                                        |      |                                |          |                                                        |
| FREE         | Segment belongs to pool of free memory objects ready for allocation by an application.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |              |         |      |                                    |      |                                                                           |      |                                                                                        |      |                                |          |                                                        |
| PRIV         | Segment contains private data.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |              |         |      |                                    |      |                                                                           |      |                                                                                        |      |                                |          |                                                        |
| SENTINAL     | Segment marks the beginning or end of the global heap.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |              |         |      |                                    |      |                                                                           |      |                                                                                        |      |                                |          |                                                        |
| owner        | Specifies the module name of the application or library that allocated the memory object. The acronym PDB is used for memory objects that represent process descriptor blocks. These blocks contain execution information about applications.                                                                                                                                                                                                                                                                                                                                                                              |              |         |      |                                    |      |                                                                           |      |                                                                                        |      |                                |          |                                                        |
| handle       | Specifies the handle of the global memory object.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |              |         |      |                                    |      |                                                                           |      |                                                                                        |      |                                |          |                                                        |
| flags        | Specifies D, which means the segment is movable and discardable.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |              |         |      |                                    |      |                                                                           |      |                                                                                        |      |                                |          |                                                        |
| chain        | Specifies the previous and next addresses in the LRU list.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |              |         |      |                                    |      |                                                                           |      |                                                                                        |      |                                |          |                                                        |

`.reboot` WDEB386 command

`.reboot`

The `.reboot` command causes the target system to restart.

## bc WDEB386 command

bc list | \*

The bc command removes one or more defined breakpoints.

| Parameter | Description                                                                                                                                |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------|
| list      | Specifies any combination of integer values in the range 0 through 9. If you specify list, the debugger removes the specified breakpoints. |
| *         | Clears all breakpoints.                                                                                                                    |

### Example

The following example removes breakpoints 0, 4, and 8:

```
bc 0 4 8
```

The following example removes all breakpoints:

```
bc *
```

## bd WDEB386 command

bd list | \*

The bd command temporarily disables one or more breakpoints. To restore breakpoints disabled by the bd command, use the be (Enable Breakpoints) command.

| Parameter | Description                                                                                                                                 |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| list      | Specifies any combination of integer values in the range 0 through 9. If you specify list, the debugger disables the specified breakpoints. |
| *         | Disables all breakpoints.                                                                                                                   |

### Example

The following example disables breakpoints 0, 4, and 8:

```
bd 0 4 8
```

The following example disables all breakpoints:

```
bd *
```

## be WDEB386 command

be list | \*

The be command restores (enables) one or more breakpoints that have been temporarily disabled by a bd (Disable Breakpoints) command.

| Parameter | Description                                                                                                                                |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------|
| list      | Specifies any combination of integer values in the range 0 through 9. If you specify list, the debugger enables the specified breakpoints. |
| *         | Enables all breakpoints.                                                                                                                   |

### Example

The following example enables breakpoints 0, 4, and 8:

```
be 0 4 8
```

The following example enables all breakpoints:

```
be *
```

## bl WDEB386 command

bl

The bl command lists current information about all breakpoints created by the bp (Set Breakpoints) command.

### Example

If no breakpoints are currently defined, the debugger displays nothing. Otherwise, the breakpoint number, enabled status, breakpoint address, number of passes remaining, initial number of passes (in parentheses), and any optional debugger commands to be executed when the breakpoint is reached are displayed on the screen, as in the following example:

```
0 e 04BA:0100
4 d 04BA:0503 4 (10)
8 e 0D2D:0001 3 (3) "R;DB DS:SI"
9 e xxxx:0012
```

In this example, breakpoints 0 and 8 are enabled (e) and 4 is disabled (d). Breakpoint 4 had an initial pass count of 10h and has four remaining passes to be taken before the breakpoint. Breakpoint 8 had an initial pass count of 3 and must make all three passes before it halts execution and forces the debugger to execute the optional debugger commands enclosed in quotation marks. Breakpoint 0 shows no initial pass count, which means it was set to 1. Breakpoint 9 shows a virtual breakpoint (a breakpoint set in a segment that has not been loaded into memory).

## bp WDEB386 command

```
bp[number]addr [count] ["cmds"]
```

The bp command creates a software breakpoint at an address. When the application is running, software breakpoints stop execution and force the debugger to execute the default or optional command string. Unlike breakpoints created by the g (Go) command, software breakpoints remain in memory until you remove them with the bc (Clear Breakpoints) command or temporarily disable them with the bd (Disable Breakpoints) command.

The debugger allows up to 10 software breakpoints (0 through 9). If you specify more than 10 breakpoints, the debugger returns the following message:

```
Too Many Breakpoints
```

The addr parameter is required for all new breakpoints.

| Parameter | Description                                                                                                                                                                                                                              |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| number    | Specifies which breakpoint is being created. No space is allowed between the bp and number. If number is omitted, the first available breakpoint number is used.                                                                         |
| addr      | Specifies any valid instruction address--the first byte of an operation code (opcode).                                                                                                                                                   |
| count     | Specifies the number of times the breakpoint is to be ignored before being executed. It can be any 16-bit value.                                                                                                                         |
| cmds      | Specifies an optional list of debugger commands to be executed in place of the default command when the breakpoint is reached. You must enclose optional commands in quotation marks and separate optional commands with semicolons (;). |

### Example

The following example creates a breakpoint at address CS:123:

```
bp 123
```

The following example creates breakpoint 8 at address 400:23 and executes a db (Display Bytes) command:

```
bp8 400:23 "db DS:SI"
```

The following example creates a breakpoint at address 100 in the current CS selector and displays the registers before comparing a block of memory. The breakpoint is ignored 16 (10h) times before being executed.

```
bp 100 10 "r;c100 L 100 300"
```



## br WDEB386 command

br[number] flags [count] ["cmds"]

The br command sets an 80386 debug register breakpoint. Debug registers can be used to break on data reads and writes and instruction execution. Up to four debug registers can be set and enabled at one time.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |   |                                    |   |                                   |   |                                               |   |                                                           |   |                       |   |                            |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---|------------------------------------|---|-----------------------------------|---|-----------------------------------------------|---|-----------------------------------------------------------|---|-----------------------|---|----------------------------|
| number    | Specifies which breakpoint is being created. No space is allowed between the br command and the number parameter. If number is omitted, the first available breakpoint number is used.                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |   |                                    |   |                                   |   |                                               |   |                                                           |   |                       |   |                            |
| flags     | Specifies the length and break conditions for the breakpoint. This parameter can be some combination of the following values:<br><table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>1</td><td>Set 1-byte length (default value).</td></tr><tr><td>2</td><td>Set word length on word boundary.</td></tr><tr><td>4</td><td>Set doubleword length on doubleword boundary.</td></tr><tr><td>E</td><td>Break on instruction execution only (1-byte length only).</td></tr><tr><td>W</td><td>Break on writes only.</td></tr><tr><td>R</td><td>Break on reads and writes.</td></tr></tbody></table> | Value | Meaning | 1 | Set 1-byte length (default value). | 2 | Set word length on word boundary. | 4 | Set doubleword length on doubleword boundary. | E | Break on instruction execution only (1-byte length only). | W | Break on writes only. | R | Break on reads and writes. |
| Value     | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |   |                                    |   |                                   |   |                                               |   |                                                           |   |                       |   |                            |
| 1         | Set 1-byte length (default value).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |   |                                    |   |                                   |   |                                               |   |                                                           |   |                       |   |                            |
| 2         | Set word length on word boundary.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |   |                                    |   |                                   |   |                                               |   |                                                           |   |                       |   |                            |
| 4         | Set doubleword length on doubleword boundary.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |   |                                    |   |                                   |   |                                               |   |                                                           |   |                       |   |                            |
| E         | Break on instruction execution only (1-byte length only).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |   |                                    |   |                                   |   |                                               |   |                                                           |   |                       |   |                            |
| W         | Break on writes only.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |   |                                    |   |                                   |   |                                               |   |                                                           |   |                       |   |                            |
| R         | Break on reads and writes.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |   |                                    |   |                                   |   |                                               |   |                                                           |   |                       |   |                            |
| count     | Specifies the number of times the breakpoint is to be ignored before being executed. It can be any 16-bit value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |   |                                    |   |                                   |   |                                               |   |                                                           |   |                       |   |                            |
| cmds      | Specifies an optional list of debugger commands to be executed in place of the default command when the breakpoint is reached. You must enclose the group of optional commands in quotation marks and separate optional commands with semicolons (;).                                                                                                                                                                                                                                                                                                                                                                 |       |         |   |                                    |   |                                   |   |                                               |   |                                                           |   |                       |   |                            |

## c WDEB386 command

c range addr

The c command compares one memory location with another memory location.

If the two memory areas are identical, the debugger displays nothing and returns the debugger prompt. Differences, when they exist, are displayed in the following form:

addr1 byte1 byte2 addr2

| Parameter | Description                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------|
| range     | Specifies the block of memory that is to be compared with a block of memory starting at addr. |
| addr      | Specifies the starting address of the second block of memory.                                 |

### Example

This section shows two forms of the c command that have the same effect. Each compares the block of memory from 100h to 1FFh with the block of memory from 300h to 3FFh.

The first example specifies a range with a starting address of 100h and an ending address of 1FFh. This block of memory is compared with a block of memory of the same size starting at 300h.

```
c100 1FF 300
```

The second example compares the same block of memory but specifies the range by using the L (length) option.

```
c100 L 100 300
```

## d WDEB386 command

d [range]

The d command displays the contents of memory at a given address or in a range of addresses. The d command displays one or more lines, depending on the range given. Each line displays the address of the first item displayed. The command always displays at least one value. The memory display is in the format defined by a previously executed db (Display Bytes), dd (Display Doublewords), or dw (Display Words) command. Each subsequent d (typed without parameters) displays the bytes immediately following those last displayed.

| Parameter | Description                                                                                                                                                                                                           |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| range     | Specifies the block of memory to display. If you omit range, the d command displays the next byte of memory after the last one displayed. The d command must be separated by at least one space from any range value. |

### Example

The following example displays 20h bytes at CS:100:

```
d CS:100 L 20
```

The following example displays all the bytes in the range 100h to 115h in the CS selector:

```
d CS:100 115
```

## db WDEB386 command

db [range]

The db command displays the values of the bytes at a given address or in a given range.

The display is in two portions: a hexadecimal display (each byte is shown in hexadecimal format) and an ASCII display (the bytes are shown as ASCII characters). A nonprinting character is denoted by a period (.) in the ASCII portion of the display. Each display line shows 16 bytes, with a hyphen between the eighth and ninth bytes. Each displayed line begins on a 16-byte boundary.

| Parameter | Description                                                                                                                                                                 |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| range     | Specifies the block of memory to display. If you omit range, 128 bytes are displayed beginning at the first address after the address displayed by the previous db command. |

### Example

The following example displays 0Ah bytes of memory, beginning at the specified address:

```
db CS:100 0A
```

This example displays lines in a format similar to the following:

```
04BA:0100 54 4F 4D 20 53 . . . 45 52 TOM SAWYER
```

Each line of the display begins with an address, incremented by 10h from the address on the previous line.

## dd WDEB386 command

dd [range]

The dd command displays the hexadecimal values of the doublewords at the address specified or in the specified range of addresses.

The dd command displays one or more lines, depending on the range given. Each line displays the address of the first doubleword in the line, followed by up to four hexadecimal doubleword values. The hexadecimal values are separated by spaces. The dd command displays values up to the end of the range or until the first 32 doublewords have been displayed.

Typing dd displays 32 doublewords at the current dump address. For example, if the last byte in the previous dd command was 04BA:0110, the display starts at 04BA:0111.

| Parameter | Description                                                                                                                                                                            |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| range     | Specifies the block of memory to display. If you omit range, 32 doubleword values are displayed beginning at the first address after the address displayed by the previous dd command. |

### Example

The following example displays the doubleword values from CS:100 to CS:110:

```
dd CS:100 110
```

The resulting display is similar to the following:

```
04BA:0100 7473:2041 676E:6972 5405:0104 0A0D:7865  
04BA:0110 0000:002E
```

No more than four values per line are displayed.

## dg WDEB386 command

dg[a] [range]

The dg command displays the specified range of entries in the global descriptor table (GDT).

| Parameter | Description                                                                                                                                                                                                                                 |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| range     | Specifies the range of entries in the GDT. If you omit range, the debugger displays the entire contents of the GDT.                                                                                                                         |
| a         | Causes all entries in the table to be displayed, not just the valid entries. By default, only the valid GDT entries are displayed. If the command is passed a local descriptor table (LDT) selector, it displays the appropriate LDT entry. |

### Example

The following example displays only the valid entries from 0h to 40h in the GDT:

```
dg 0 40
```

The resulting display is similar to the following:

```
0008 Data Seg Base=01D700 Limit=3677 DPL=0 Present ReadWriteAccessed
0010 TSS Desc Base=007688 Limit=002B DPL=0 Present Busy
0018 Data Seg Base=020D7A Limit=03FF DPL=0 Present ReadWrite
0020 Data Seg Base=000000 Limit=03FF DPL=0 Present ReadWrite
0028 LDT Desc Base=000000 Limit=0000 DPL=0 Present
0030 Data Seg Base=000000 Limit=0000 DPL=0 Present ReadWrite
0040 Data Seg Base=000400 Limit=03BF DPL=3 Present ReadWrite
```

## di WDEB386 command

di[a] [range]

The di command displays the specified range of entries in the interrupt descriptor table (IDT).

| Parameter | Description                                                                                                                     |
|-----------|---------------------------------------------------------------------------------------------------------------------------------|
| a         | Causes all entries in the table to be displayed, not just the valid ones. The default is to display just the valid IDT entries. |
| range     | Specifies the range of entries to be displayed. If you omit range, the debugger displays all IDT entries.                       |

### Example

The following example displays the valid IDT entries in the range 0h through 10h:

```
di 0 10
```

The resulting display is similar to the following:

```
0000 Int Gate Sel=1418 Offst=03D8 DPL=3 Present
0001 Int Gate Sel=2D38 Offst=0049 DPL=3 Present
0002 Int Gate Sel=1418 Offst=03E4 DPL=3 Present
0003 Int Gate Sel=2D38 Offst=006F DPL=3 Present
0004 Int Gate Sel=1418 Offst=0417 DPL=3 Present
0005 Int Gate Sel=1418 Offst=041D DPL=3 Present
0006 Int Gate Sel=1418 Offst=0423 DPL=3 Present
0007 Int Gate Sel=2D38 Offst=00A3 DPL=3 Present
0008 Int Gate Sel=1418 Offst=042F DPL=3 Present
0009 Int Gate Sel=2D38 Offst=00CA DPL=3 Present
000A Int Gate Sel=2D38 Offst=00D3 DPL=3 Present
000B Int Gate Sel=2D38 Offst=0156 DPL=3 Present
000C Int Gate Sel=2D38 Offst=01A4 DPL=3 Present
000D Int Gate Sel=2D38 Offst=01C6 DPL=3 Present
```

## dl WDEB386 command

dl[a | p | s | h] [range]

The dl command displays the specified range of entries in the local descriptor table (LDT).

| Parameter | Description                                                                                                                                                                                                                               |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| a         | Causes all entries in the table to be displayed, not just the valid ones. By default, only the valid LDT entries are displayed. If the command is passed a global descriptor table (GDT) selector, it displays the appropriate GDT entry. |
| p         | Causes private segment selectors to be displayed.                                                                                                                                                                                         |
| s         | Causes shared segment selectors to be displayed.                                                                                                                                                                                          |
| h         | Causes huge segment selectors to be displayed. To display the huge segment selectors, give the shadow selector followed by the maximum number of selectors reserved for that segment plus 1.                                              |
| range     | Specifies the range of entries to be displayed. If you omit range, the entire table is displayed.                                                                                                                                         |

### Example

The following example displays all the LDT entries:

```
dl a 4 57
```

The command produces a display similar to the following:

```
0014 Call Gate Sel=1418 Offst=0417 DPL=0 NotPres WordCount=1D
001C Code Seg Base=051418 Limit=0423 DPL=0 NotPres ExecOnly
0027 Reserved Base=87F000 Limit=FEA5 DPL=3 Present
0034 Code Seg Base=05F000 Limit=1805 DPL=0 NotPres ExecOnly
003C Code Seg Base=05F000 Limit=EF57 DPL=0 NotPres ExecOnly
0047 Code Seg Base=4DC000 Limit=0050 DPL=3 Present ExecOnly
004D Reserved Base=71F000 Limit=F841 DPL=1 NotPres
0057 Code Seg Base=59F000 Limit=E739 DPL=3 Present ExecOnly
```



## dp WDEB386 command

dp[a|d] [range]

The dp command displays the page directory and page tables. Page tables are always skipped if the corresponding page directory entry is not present. Page directory entries appear with an asterisk next to the page frame.

| Parameter | Description                                                                                                                                      |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| a         | Displays all present page directory and page table entries; by default, page directory and page table entries that are zero are skipped.         |
| d         | Displays only page directory entries. If a count is given as part of the optional range, it will be interpreted as a page directory entry count. |
| range     | Specifies the range of linear addresses for page tables.                                                                                         |

### Example

The following example displays the page directory and page table in the range 0 through 12h:

```
dp 0 12
```

The resulting display is similar to the following:

```
%00000000 *frame=00FCE state=3 res=0 c A pb1=0 pb0=0 U W P
%00000000 frame=00000 state=3 res=0 c u pb1=0 pb0=0 U W P
%00001000 frame=00001 state=3 res=0 c u pb1=0 pb0=0 U W P
```

The display produced by the dp command can contain flags that have the following meanings:

| Bit set | Bit clear | Meaning             |
|---------|-----------|---------------------|
| D       | c         | Dirty/clean         |
| A       | u         | Accessed/unaccessed |
| U       | s         | User/supervisor     |
| W       | r         | Writable/read-only  |
| P       | n         | Present/not-present |

## dt WDEB386 command

dt [addr]

The dt command displays the current task state segment (TSS) or the selected TSS if you specify the optional address.

| Parameter | Description                                                                                                                  |
|-----------|------------------------------------------------------------------------------------------------------------------------------|
| addr      | Specifies the address of the TSS to display. If no addr is given, dt displays the current TSS pointed to by the TR register. |

### Example

The following example displays the current TSS:

```
dt
```

The resulting display is similar to the following:

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000  
IP=0000 CS=0000 DS=0000 ES=0000 SS=0000 NV UP DI PL NZ NAPO NC  
SS0=0038 SP0=08DE SS1=0000 SP1=0000 SS2=0000 SP2=0000  
IOPL=0 LDTR=0028 LINK=0000
```

## dw WDEB386 command

dw [range]

The dw command displays the hexadecimal values of the words at a given address or in a given range of addresses.

The command displays one or more lines, depending on the range given. Each line displays the address of the first word in the line, followed by up to eight hexadecimal word values. The hexadecimal values are separated by spaces. The command displays values until the end of the range or until the first 64 words have been displayed.

Typing dw displays 64 words at the current dump address. For example, if the last word in the previous dw command was displayed at address 04BA:0110, the next display will start at 04BA:0112.

| Parameter | Description                                                                                                                                                                   |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| range     | Specifies the range of addresses to display. If you omit range, 64 words are displayed beginning at the first address after the address displayed by the previous dw command. |

### Example

The following example displays the word values from CS:100 to CS:110:

```
dw CS:100 110
```

The resulting display is similar to the following:

```
04BA:0100 2041 7473 6972 676E 0104 5404 7865 0A0D  
04BA:0110 002E
```

## e WDEB386 command

e addr [list]

The e command enters byte values into memory at a specified address. You can specify the new values on the command line or let the debugger prompt you for values. If the debugger prompts you, it displays the address and its contents and then waits for you to perform one of the following actions:

- Replace a byte value with a value you type. Type the value after the current value. If the byte you type is an invalid hexadecimal value or contains more than two digits, the system does not echo the illegal or extra character.
- Press the SPACEBAR to advance to the next byte. To change the value, type the new value after the current value. If, when you press the SPACEBAR, you move beyond an 8-byte boundary, 80386 Debugger starts a new display line with the address displayed at the beginning.
- Type a hyphen (-) to return to the preceding byte. If you decide to change a byte before the current position, typing the hyphen returns the current position to the previous byte. When you type the hyphen, a new line is started with its address and byte value displayed.
- Press ENTER to terminate the e command. You can press ENTER at any byte position.

| Parameter | Description                                                                                                                                                                              |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| addr      | Specifies the address of the first byte to be entered.                                                                                                                                   |
| list      | Specifies the byte values used for replacement. These values are inserted automatically. If an error occurs when you are using the list form of the command, no byte values are changed. |

### Example

The following example prompts you to change the value EB at CS:100:

```
eCS:100  
04BA:0100 EB.
```

To step through the subsequent bytes without changing values, press the SPACEBAR. In the following example, the SPACEBAR is pressed three times:

```
04BA:0100 EB.41 10. 00. BC.
```

To return to a value at a previous address, type a hyphen, as shown in the following example:

```
04BA:0100 EB.41 10. 00. BC.-  
04BA:0102 00.-  
04BA:0101 10.
```

This example returns to the address CS:101.

## f WDEB386 command

f range list

The f command fills the addresses in a specified range with the values in the specified list.

| Parameter | Description                                                                                                                                                                                                                                                                                      |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| range     | Specifies the block of memory to be filled. If range contains more bytes than the number of values in list, the debugger uses list repeatedly until all bytes in range are filled. If any of the memory in range is not valid (bad or nonexistent), an error occurs in all succeeding locations. |
| list      | Specifies the list of values to fill the given range. If list contains more values than the number of bytes in range, the debugger ignores the extra values in list.                                                                                                                             |

### Example

The following example fills memory locations 04BA:100 through 04BA:1FF with the bytes specified, repeating the five values until it has filled all 100h bytes:

```
f04BA:100 L 100 42 45 52 54 41
```

## g WDEB386 command

`g[s|h|t|z] [=addr [addr[...]] ]`

The `g` command executes the application currently in memory. If you type the `g` command by itself, the current application runs as if it had been run outside the debugger. If you specify `=addr`, execution begins at the specified address.

Specifying an optional breakpoint address causes execution to halt at the first address encountered, regardless of the position of the address in the list of addresses that halts execution or application branching. When execution of the application reaches a breakpoint, the default command string is executed.

The stack (SS:SP) must be valid and have 6 bytes available for this command. The `g` command uses an `iret` instruction to cause a jump to the application being tested. The stack is set, and the user flags, CS register, and IP register are pushed on the user stack. (If the user stack is not valid or is too small, the operating system may crash.) An interrupt code (0CCh) is placed at the specified breakpoint addresses.

When the debugger encounters an instruction with the breakpoint code, it restores all breakpoint addresses listed with the `g` command to their original instructions. If you do not halt execution at one of the breakpoints, the interrupt codes are not replaced with the original instructions.

| Parameter                        | Description                                                                                                                                                                                                                                                                |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>s</code>                   | Shows the time, in microseconds, from when the system is started with <code>gs</code> until the next entry to the debugger. No attempt is made to calculate and remove debugger overhead from the measurement. Requires a timing card.                                     |
| <code>h</code>                   | Displays the approximate debugger overhead in the <code>s</code> option. Requires a timing card.                                                                                                                                                                           |
| <code>t</code> or <code>z</code> | Allows trapped exceptions to resume at the original trap handler address without having to unhook the exception. Use these options instead of the <code>vcp d; t; vsp d</code> command.                                                                                    |
| <code>=addr</code>               | Specifies the address at which execution is to begin. The equal sign (=) is needed to distinguish the starting address from the breakpoint address.                                                                                                                        |
| <code>addr</code>                | Specifies one or more breakpoint addresses where execution is to halt. You can specify up to 10 breakpoints, but only at addresses containing the first byte of an operation code (opcode). If you attempt to set more than 10 breakpoints, an error message is displayed. |

### Example

The following example executes the application currently in memory until address 7550 in the CS selector is executed. The debugger then executes the default command string, removes the `int 3` trap from this address, and restores the original instruction. When you resume execution, the original instruction is executed.

```
gCS:7550
```

## h WDEB386 command

h word word

The h command performs hexadecimal arithmetic on the two specified parameters.

The debugger adds, subtracts, and multiplies the two parameters; divides the second parameter by the first; and then displays the results on one line. The debugger does 32-bit multiplication and displays the result as doublewords. The debugger displays the result of division as a 16-bit quotient and a 16-bit remainder.

| Parameter | Description                        |
|-----------|------------------------------------|
| word      | Specifies a 16-bit word parameter. |

### Example

The following example performs the calculations on 300h and 100h:

```
h 300 100
```

The resulting display is the following:

```
+0400 -0200 *0000 0003 /0003 0000
```

## i WDEB386 command

i word

The i command accepts and displays 1 byte from a specified port.

| Parameter | Description                        |
|-----------|------------------------------------|
| word      | Specifies the 16-bit port address. |

### Example

The following example displays the byte at port address 2F8h:

```
i 2F8
```



## j WDEB386 command

j expr ["cmds"]

The j command executes the specified commands when the specified expression is TRUE. If expr is FALSE, the debugger continues to the next command line (excluding the commands in cmds).

The j command is useful in breakpoint commands to conditionally break execution when an expression becomes TRUE.

| Parameter | Description                                                                                                                                                                                                                                     |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| expr      | Evaluates to a Boolean TRUE or FALSE.                                                                                                                                                                                                           |
| cmds      | Specifies a list of debugger commands to be executed when expr is TRUE. The list must be enclosed in single or double quotation marks. You must separate optional commands with semicolons (;). Single commands do not require quotation marks. |

### Example

The following example causes execution to break if AX does not equal zero when the breakpoint is reached:

```
bp 167:1454 "J AX == 0;G"
```

The following example displays the registers and continues execution when the byte pointed to by DS:SI + 3 is equal to 40h; otherwise, it displays the descriptor table:

```
bp 167:1462 "J BY (DS:SI+3) == 40 'R;G';DG DS"
```

## k WDEB386 command

k[b|s|v] [addr] [addr]

This command displays the current stack frame. Each line shows the name of a procedure, its arguments, and the address of the statement that called it. The command displays four 2-byte arguments by default. The ka command changes the number of arguments displayed by this command.

Using the k command at the beginning of a function (before the function prolog has been executed) gives incorrect results. The command uses the BP register to compute the current backtrace, and this register is not correctly set for a function until its prolog has been executed.

| Parameter | Description                                                                                                                           |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------|
| b         | Indicates the stack frame is 32 bits wide.                                                                                            |
| s         | Indicates the stack frame is 16 bits wide.                                                                                            |
| v         | Displays the verbose version of stack information--that is, information about stack location and frame pointer values for each frame. |
| addr      | Specifies an optional stack-frame address (SS:BP) or an optional code address (CS:IP).                                                |

## ka WDEB386 command

ka count

The ka command sets the number of arguments displayed for all subsequent stack trace commands. The initial default value is 4.

| Parameter | Description                                                                                                |
|-----------|------------------------------------------------------------------------------------------------------------|
| count     | Specifies the number of arguments to be displayed. The count parameter must be in the range 0 through 1Fh. |

## kt WDEB386 command

k[b|s|v]t [addr]

This command displays the stack frame of the current task or the task specified by the addr parameter. Each line shows the name of a procedure, its arguments, and the address of the statement that called it. The command displays four 2-byte arguments by default. The ka command changes the number of arguments displayed by this command.

| Parameter | Description                                                                                                                                                                                                                                           |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| b         | Indicates the stack frame is 32 bits wide.                                                                                                                                                                                                            |
| s         | Indicates the stack frame is 16 bits wide.                                                                                                                                                                                                            |
| v         | Displays the verbose version of stack information--that is, information about stack location and frame pointer values for each frame.                                                                                                                 |
| addr      | Specifies the segment address of the process descriptor block (PDB) for the task to be traced. To obtain the addr value, use the .dq (Dump Task Queue) command. If addr is not supplied, the kt command displays the stack frame of the current task. |

la WDEB386 command

la

The la command lists the absolute symbols in the active map.

## lg WDEB386 command

lg

The lg command lists the selector (or segment) and the name of each group in the active map.

Example

The lg command produces a display similar to the following:

```
#0090:0000 DOSCODE
#0828:0000 DOSGROUP
#1290:0000 DBGCODE
#16C0:0000 DBGDATA
#1A38:0000 TASKCODE
#1AD8:0000 DOSRING3CODE
#1AE0:0000 DOSINITCODE
#2018:0000 DOSINITRMCODE
#20A8:0000 DOSINITDATA
#23F8:0000 DOSMTE
#2420:0000 DOSHIGHDATA
#28D0:0000 DOSHIGHCODE
#3628:0000 DOSHIGH2CODE
#0090:0000 DOSCODE
```

## lm WDEB386 command

lm

The lm command lists the symbol files currently loaded and indicates which one is active.

The last symbol file loaded is made active by default. Use the w (Change Map) command to change the active file.

Example

The lm command returns a display similar to the following:

```
COMSAM2D is active.  
DISK01D.
```

## In WDEB386 command

In [addr]

The In command lists the symbol nearest the specified address. The command lists the nearest symbol before and after the specified addr parameter. This command also shows line-number information if it is available in the symbol file.

| Parameter | Description                                                                                    |
|-----------|------------------------------------------------------------------------------------------------|
| addr      | Specifies any valid instruction address. The default value is the current disassembly address. |

### Example

The In command without the addr parameter displays the nearest symbols before and after the current disassembly address. The output looks similar to the following:

```
6787 VerifyRamSemAddr + 10  
67AA PutRamSemID - 13
```



## ls WDEB386 command

ls group-name | name-chars | \*

The ls command lists the symbols in the specified group or lists names that match the search specification in all groups. The only valid wildcard is a single asterisk (\*) as the last character on the command line; all other characters are ignored.

| Parameter  | Description                                                         |
|------------|---------------------------------------------------------------------|
| group-name | Names the group that contains the symbols you want to list.         |
| name-chars | Specifies the beginning characters of the symbols you want to list. |

### Example

The following example displays all the symbols in the DOSRING3CODE group:

```
ls DOSRING3CODE
```

Symbols are displayed in a format similar to the following:

```
0000 Sigdispatch
001A LibInitDisp
```

The following example displays all the symbols that begin with the string vkd:

```
ls vkd*
```

Group names are displayed as they are searched, in a form similar to the following:

```
GROUP: [0028] CODE
        60003A74 VKD_Control_Debug
GROUP: [0030] DATA
        6001DFFC VKD_CB_Offset
GROUP: [0030} IDATA
```

The following example displays the address and group for the symbol VMM\_base:

```
ls vmm_base
```

## m WDEB386 command

m range addr

The m command moves a block of memory from one memory location to another.

Overlapping moves--those in which part of the block overlaps some of the current addresses--are always performed without loss of data. Addresses that could be overwritten are moved first. For moves from higher to lower addresses, the sequence of events is first to move the data at the block's lowest address and then to work toward the highest. For moves from lower to higher addresses, the sequence is first to move the data at the block's highest address and then to work toward the lowest.

Note that if the addresses in the block being moved will not have new data written to them, the data that was in the block before the move will remain. The m command copies the data from one area into another, in the sequence described, and writes over the new addresses--hence, the importance of the moving sequence.

To review the results of a memory move, use the d (Display Memory) command, specifying the same address you used with the m command.

| Parameter | Description                                                            |
|-----------|------------------------------------------------------------------------|
| range     | Specifies the block of memory to be moved.                             |
| addr      | Specifies the starting address at which the memory is to be relocated. |

### Example

The following example first moves the data at address CS:110 to CS:510 and then moves the data at CS:10F to CS:50F, and so on, until the data at CS:100 is moved to CS:500:

```
mCS:100 110 CS:500
```

o WDEB386 command

o word byte

The o command writes a byte to a 16-bit port address.

| Parameter | Description                                          |
|-----------|------------------------------------------------------|
| word      | Specifies the 16-bit port address to be written to.  |
| byte      | Specifies the 8-bit value to be written to the port. |

Example

The following example writes the byte value 4Fh to output port 2F8h:

o 2F8 4F

## p WDEB386 command

p[n] [=addr][count]

The p command executes the instruction at a specified address and displays the current values of all the registers and flags (whatever the zd command has been set to). It then executes the default command string, if any.

The p command is identical to the t (Trace Instructions) command, except that it automatically executes and returns from any calls or software interrupts it encounters. The t command always stops after executing into the call or interrupt, leaving execution control inside the called routine.

| Parameter | Description                                                                                                                                                                                                                      |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| n         | Suppresses the register display so just the assembly line is displayed. The suppression results only if the default command, zd, is set to a normal setting, r.                                                                  |
| addr      | Specifies the starting address at which to begin execution. If you omit the optional addr parameter, execution begins at the instruction pointed to by the CS and IP registers. Use the equal sign (=) only if you specify addr. |
| count     | Specifies the number of instructions to execute before stopping and executing the default command string. The command executes the default command string for each instruction before executing the next.                        |

### Example

The following example executes the instruction pointed to by the current CS and IP register values before it executes the default command string:

```
p
```

The following example executes the instruction at address CS:120 before it executes the default command string:

```
p=120
```

## r WDEB386 command

r reg=word

The r command displays the contents of one or more central processing unit (CPU) registers and allows the contents to be changed to new values. If you specify the reg parameter with the r command, the 16-bit value of that register is displayed in hexadecimal format followed by a colon (:) prompt on the next line. You can then enter a new word value for the specified register or press ENTER if you do not want to change the register value.

If you specify f for reg, the debugger displays the flags in a row at the beginning of a new line and displays a hyphen (-) after the last flag.

You can type new flag values in any order as alphabetic pairs. You do not have to leave spaces between these values. To terminate the r command, press ENTER. Any flags for which you did not specify new values remain unchanged.

If you type more than one value for a flag or enter an invalid flag name, the flags up to the error in the list are changed and those flags at and after the error are not changed. In addition, 80386 Debugger returns the following error message:

Bad Flag

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|---------|----|--------------|----|----------------|----|---------------------|----|---------------------|----|-------------------|----|--------------------|----|---------------|----|---------------|----|----------|----|------------|----|---------------------|----|-----------------------|----|-------------|----|------------|----|-----------|----|-------------|----|---------------------------------|-----------|--------|----|---------------------------|----|---------------------------------------------|----|-------------------------------------------|----|------------------------------|
| reg       | Specifies the register to be displayed. If you omit reg, the debugger displays the contents of all registers and flags along with the next executable instruction.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| word      | Specifies the new value for the register. For the Flags register, set or clear a flag by using one of the following names:<br><table><thead><tr><th>Flag code</th><th>Meaning</th></tr></thead><tbody><tr><td>OV</td><td>Overflow set</td></tr><tr><td>NV</td><td>Overflow clear</td></tr><tr><td>DN</td><td>Direction decrement</td></tr><tr><td>UP</td><td>Direction increment</td></tr><tr><td>EI</td><td>Interrupt enabled</td></tr><tr><td>DI</td><td>Interrupt disabled</td></tr><tr><td>NG</td><td>Sign negative</td></tr><tr><td>PL</td><td>Sign positive</td></tr><tr><td>ZR</td><td>Zero set</td></tr><tr><td>NZ</td><td>Zero clear</td></tr><tr><td>AC</td><td>Auxiliary carry set</td></tr><tr><td>NA</td><td>Auxiliary carry clear</td></tr><tr><td>PE</td><td>Parity even</td></tr><tr><td>PO</td><td>Parity odd</td></tr><tr><td>CY</td><td>Carry set</td></tr><tr><td>NC</td><td>Carry clear</td></tr><tr><td>NT</td><td>Nested task switch (on and off)</td></tr></tbody></table> <p>For the machine status word (MSW) register, use the following names to set a flag:</p> <table><thead><tr><th>Flag name</th><th>Action</th></tr></thead><tbody><tr><td>TS</td><td>Sets the task switch bit.</td></tr><tr><td>EM</td><td>Sets the emulation processor extension bit.</td></tr><tr><td>MP</td><td>Sets the monitor processor extension bit.</td></tr><tr><td>PM</td><td>Sets the protected-mode bit.</td></tr></tbody></table> | Flag code | Meaning | OV | Overflow set | NV | Overflow clear | DN | Direction decrement | UP | Direction increment | EI | Interrupt enabled | DI | Interrupt disabled | NG | Sign negative | PL | Sign positive | ZR | Zero set | NZ | Zero clear | AC | Auxiliary carry set | NA | Auxiliary carry clear | PE | Parity even | PO | Parity odd | CY | Carry set | NC | Carry clear | NT | Nested task switch (on and off) | Flag name | Action | TS | Sets the task switch bit. | EM | Sets the emulation processor extension bit. | MP | Sets the monitor processor extension bit. | PM | Sets the protected-mode bit. |
| Flag code | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| OV        | Overflow set                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| NV        | Overflow clear                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| DN        | Direction decrement                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| UP        | Direction increment                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| EI        | Interrupt enabled                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| DI        | Interrupt disabled                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| NG        | Sign negative                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| PL        | Sign positive                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| ZR        | Zero set                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| NZ        | Zero clear                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| AC        | Auxiliary carry set                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| NA        | Auxiliary carry clear                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| PE        | Parity even                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| PO        | Parity odd                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| CY        | Carry set                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| NC        | Carry clear                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| NT        | Nested task switch (on and off)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| Flag name | Action                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| TS        | Sets the task switch bit.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| EM        | Sets the emulation processor extension bit.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| MP        | Sets the monitor processor extension bit.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |
| PM        | Sets the protected-mode bit.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |           |         |    |              |    |                |    |                     |    |                     |    |                   |    |                    |    |               |    |               |    |          |    |            |    |                     |    |                       |    |             |    |            |    |           |    |             |    |                                 |           |        |    |                           |    |                                             |    |                                           |    |                              |

### Comments

Setting the protected-mode bit from within the debugger does not set the target system to run in protected mode. The debugger simulates the setting. To configure the target system to run in protected mode, you would have to set the PM bit in the MSW register and reset the target system to restart in protected mode.

### Example

The r command without parameters produces a display similar to the following:

```
AX=0698 BX=2008 CX=2C18 DX=18AB SP=1B7A BP=00FF SI=0020 DI=10CD
IP=0450 CS=18B0 DS=1BE8 ES=0DA8 SS=0048 NV UP DI PL NZ NA PONC
GDTR=01BE80 3687 IDTR=01F508 03FF TR=0010 LDTR=0028 IOPL=3 MSW=PM
18B0:0450 C3 RET
```

The following example displays each flag with a two-letter code. To change any flag, type the two-letter code that inverts the setting. The flags are either set or cleared.

```
rf
```

The example produces a display similar to the following:

```
NV UP DI NG NZ AC PE NC - _
```

To change the value of a flag's setting, type the two-letter code that inverts the setting for that flag. The following example changes the sign flag to positive, enables interrupts, and sets the carry flag:

```
NV UP DI NG NZ AC PE NC - PLEICY
```

The following command modifies the MSW bits:

```
rmsw
```

Then 80386 Debugger displays the status of the MSW register and prints a colon on the next line.

## s WDEB386 command

s range list | "string"

The s command searches an address range for a specified list of bytes or an ASCII character string.

You can include one or more bytes in list, but multiple bytes must be separated by a space or comma.

When you search for more than one byte, the command returns the address of only the first byte in the string. When list contains only one byte, the debugger displays the addresses of all occurrences of the byte in range.

| Parameter | Description                                                                                             |
|-----------|---------------------------------------------------------------------------------------------------------|
| range     | Specifies the block of memory to be searched.                                                           |
| list      | Specifies one or more byte values to search for.                                                        |
| string    | Specifies an ASCII character string to be searched for. The string must be enclosed in quotation marks. |

### Example

The following example searches for byte 41h in the address range CS:100 to CS:110:

```
sCS:100 110 41
```

If it finds the value, this command produces a display similar to the following:

```
04BA:0104
```

```
04BA:010D
```

## t WDEB386 command

t[a|c|n|s|x|z][=start\_addr][count][addr]

The t command executes one or more instructions along with the default command string and then displays the decoded instruction. If you include the start\_addr parameter, tracing starts at the specified address. Otherwise, the command steps through the next machine instruction and then executes the default command string.

The t command uses the hardware trace mode of the Intel microprocessor. Consequently, you can also trace instructions stored in read-only memory (ROM).

| Parameter  | Description                                                                                                                                                         |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| a          | Indicates that an ending address is specified for the trace. Instructions are traced until the address in addr is reached.                                          |
| c          | Suppresses all output and counts instructions traced. An ending address is required for this command. Instructions are traced until the address in addr is reached. |
| n          | Suppresses the register display so just the assembly line is displayed. This works only if the default command, zd, is set to r (the normal setting).               |
| s          | Suppresses output; the instruction and count are displayed for each call and the return from that call.                                                             |
| x          | Forces the debugger to trace regions of code known to be untraceable (_PGSwitchContext, for example).                                                               |
| z          | Allows original trap handler address to be traced into without having to unhook the exception. Use this option instead of vcp d; t; vsp d.                          |
| start_addr | Specifies the instruction address at which to start tracing. The equal sign (=) is required.                                                                        |
| count      | Specifies the number of instructions to execute and trace.                                                                                                          |
| addr       | Specifies the instruction address at which to stop tracing.                                                                                                         |

### Example

The following example traces the current position (04BA:011A) and uses the default command string (r command) to display registers:

t

The resulting output is similar to the following:

```
AX=0E00  BX=00FF  CX=0007  DX=01FF  SP=039D  BP=0000  SI=005C  DI=0000
IP=011A  CS=04BA  DS=04BA  ES=04BA  SS=04BA  NV UP DI NG NZ AC PENC
GDTR=01D700 3677  IDTR=020D7A 03FF  TR=0010  LDTR=0028 IOPL=3 MSW=PM
04BA:011A  CD21PUSH  21
```

The following command causes the debugger to execute 16 (10h) instructions beginning at 011A in the current selector:

t=011A 10

The debugger executes and displays the results of the default command string for each instruction. The display is scrolled until the last instruction is executed. Press the CTRL+S key combination to stop the scrolling and CTRL+Q to resume.



## u WDEB386 command

u [range]

The u command disassembles bytes and displays the source statements, with addresses and byte values, that correspond to them.

The display of disassembled code looks similar to a code listing for an assembled file. If you type the u command by itself, 20h bytes are disassembled at the first address after the one displayed by the previous u command.

| Parameter | Description                                                                                                                                    |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------|
| range     | Specifies the block of memory in which instructions are to be disassembled. If no range is given, the command disassembles the next 20h bytes. |

### Example

The following example disassembles and displays 20h bytes from the specified address:

```
uCS:046C
```

The resulting display is similar to the following:

```
1A60:046C C3RET
1A60:046D 9A6B3E100D CALL 0D10:3E6B
1A60:0472 33C0 XOR AX,AX
1A60:0474 50PUSH AX
1A60:0475 9DPOPF
1A60:0476 9CPUSHF
1A60:0477 58POP AX
1A60:0478 2500F0 AND AX,F000
1A60:047B 3D00F0 CMP AX,F000
1A60:047E 7508 JNZ 0488
1A60:0480 689C26 PUSH 269C
1A60:0483 9AF105100D CALL 0D10:05F1
```

If the bytes at some addresses are altered, the disassembler alters the instruction statements. You can also use the u command for the changed locations, for the new instructions viewed, and for the disassembled code used to edit the source file.

v WDEB386 command

v

The v command displays the current 80386 Debugger version number and date.

## vc WDEB386 command

vc[n | p | r | v] number[,number [...]]

The vc command clears the specified interrupt vector and reinstalls the previous interrupt vector.

| Parameter | Description                                                                       |
|-----------|-----------------------------------------------------------------------------------|
| n         | Removes the beep from traps that beep when encountered; does not clear the traps. |
| p         | Clears protected-mode vectors only.                                               |
| r         | Clears real-mode vectors only.                                                    |
| v         | Clears virtual 8086 (V86) mode vectors only.                                      |
| number    | Specifies the interrupt vector to clear.                                          |

## vl WDEB386 command

vl[n | p | r | v]

Lists the interrupt vectors that the debugger intercepts. Vectors that have been set with the vt command (as opposed to vs) are listed with an asterisk (\*) following the vector number.

| Parameter | Description                                     |
|-----------|-------------------------------------------------|
| n         | Lists the traps that beep when encountered.     |
| p         | Lists the protected-mode vectors only.          |
| r         | Lists the real-mode vectors only.               |
| v         | Lists the virtual 8086 (V86) mode vectors only. |

## vo WDEB386 command

vo[n | p | r | v]

The vo command lists interrupt vectors in the display format based on the newvec option. For details, see the y command.

| Parameter | Description                                     |
|-----------|-------------------------------------------------|
| n         | Lists the traps that beep when encountered.     |
| p         | Lists the protected-mode vectors only.          |
| r         | Lists the real-mode vectors only.               |
| v         | Lists the virtual 8086 (V86) mode vectors only. |

## vs WDEB386 command

vs[n | p | r | v] number[,number[,...]]

The vs command adds a new interrupt vector to the list of intercepted vectors. Vectors set by this command do not intercept interrupts that occur at ring 0.

| Parameter | Description                                     |
|-----------|-------------------------------------------------|
| n         | Lists the traps that beep when encountered.     |
| p         | Lists the protected-mode vectors only.          |
| r         | Lists the real-mode vectors only.               |
| v         | Lists the virtual 8086 (V86) mode vectors only. |
| number    | Specifies the interrupt vector to intercept.    |

## vt WDEB386 command

vt[n | p | r | v] number[,number[,...]]

The vt command adds a new interrupt vector to the list of intercepted vectors.

| Parameter | Description                                     |
|-----------|-------------------------------------------------|
| n         | Lists the traps that beep when encountered.     |
| p         | Lists the protected-mode vectors only.          |
| r         | Lists the real-mode vectors only.               |
| v         | Lists the virtual 8086 (V86) mode vectors only. |
| number    | Specifies the interrupt vector to intercept.    |

## w WDEB386 command

w [map-name]

The w command changes the active map file.

| Parameter | Description                                                                                                                                                                                                                                                                       |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| map-name  | Specifies the name of the map file you want to make active. Use the lm (List Map) command to display a list of available map files.<br>If map-name is not specified, the loaded maps are displayed and the user is prompted to select a map by pressing its corresponding number. |

### Example

The lm command can be used to display the loaded map files in a form similar to the following:

```
COMSAM2D is active.  
DISK01D.
```

Then the following command can be used to change the active map file to DISK01D:

```
w DISK01D
```

The following command displays the list of loaded maps:

```
w
```

The resulting display is similar to the following, prompting the user to type the number corresponding to the map to activate:

```
1. KERNEL  
2. Win386 is active  
activate which map?
```

In this case, pressing 1 activates the KERNEL map; pressing 2 leaves the Win386 map activated; and pressing the SPACEBAR leaves the current map activated. Any other key is ignored, and the debugger will continue to wait for input.



## wa WDEB386 command

wa map-name

The wa command adds the specified map to the list of active maps.

| Parameter | Description                                          |
|-----------|------------------------------------------------------|
| map-name  | Specifies the map to add to the list of active maps. |

## wr WDEB386 command

wr map-name

The wr command removes the specified map from the list of active maps.

| Parameter | Description                                               |
|-----------|-----------------------------------------------------------|
| map-name  | Specifies the map to remove from the list of active maps. |

## y WDEB386 command

y[? | option]

The y command changes the debugger configuration. The following list describes the available configuration options. All settings are toggles.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                            |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?         | Displays a list of supported options.                                                                                                                                                                                                                                                                                                                                  |
| option    | Following are the available configuration options:                                                                                                                                                                                                                                                                                                                     |
| /a        | Controls automatic symbol loading. If this option is set, Windows will not load symbols automatically.                                                                                                                                                                                                                                                                 |
| /n        | Sets the following options:<br><br>codebytes<br>dislwr<br>int3line<br>newprompt<br>newreg<br>newvec<br>symaddr                                                                                                                                                                                                                                                         |
| /v        | Controls segment load notification messages. If this option is set, all segment load notifications will be displayed.                                                                                                                                                                                                                                                  |
| 386env    | Controls the size of addresses, registers, and so on when displayed. When this option is on, addresses, registers, and so on are shown in 32-bit format; otherwise, they are shown in 16-bit format.                                                                                                                                                                   |
| codebytes | Causes the disassembler to display the code bytes along with the disassembled instructions.                                                                                                                                                                                                                                                                            |
| disaddr   | Causes the disassembler to display the disassembly address.                                                                                                                                                                                                                                                                                                            |
| disline   | Causes the disassembler to display the filename and line number of each operation code (opcode).                                                                                                                                                                                                                                                                       |
| dislwr    | Controls the disassembler's lowercase option. When the flag is on, disassembly is in lowercase.                                                                                                                                                                                                                                                                        |
| int3line  | Causes the disassembler to display the filename and line number on int 3 instructions.                                                                                                                                                                                                                                                                                 |
| newprompt | Causes 80386 Debugger to produce a double prompt when paging is enabled and a nesting level if the debugger is reentered.                                                                                                                                                                                                                                              |
| newreg    | Controls the format of the register display.                                                                                                                                                                                                                                                                                                                           |
| newvec    | Controls the display format for the intercepted interrupt vectors.                                                                                                                                                                                                                                                                                                     |
| regterse  | Controls the number of registers displayed by the r (Register Dump) command. In the 80386 environment, when regterse is on, only the first three lines are displayed (instead of the normal six lines plus disassembly line). In the 80286 environment (386env off), only the first two lines are displayed (instead of the normal three lines plus disassembly line). |
| scrncols  | Sets the number of screen columns in the debug display. The default is 79 columns.                                                                                                                                                                                                                                                                                     |
| scrnlines | Sets the number of screen lines in the debug display. The default is 24 lines.                                                                                                                                                                                                                                                                                         |
| skipint3s | Causes the debugger to ignore inline int 3 instructions.                                                                                                                                                                                                                                                                                                               |
| symaddr   | Causes the disassembler to display symbol values along with the symbols.                                                                                                                                                                                                                                                                                               |
| teftibase | Sets the base port address for the timing card.                                                                                                                                                                                                                                                                                                                        |

## z WDEB386 command

z

Replaces the instruction bytes of the current int 3 instruction or the previous int 1 instruction with nop instructions. This allows the user to avoid int 1 or int 3 instructions that were assembled into the executable file by breaking into the debugger more than once.

## zd WDEB386 command

zd

The zd command executes the default command string.

The default command string is initially set to the r (Display Registers) command by the debugger. The default command string is executed every time a breakpoint is encountered during execution of the application or whenever a p (Program Trace) or t (Trace Instructions) command is executed.

Use the zl command to display the default command string and the zs command to change the default command string.

zl WDEB386 command

zl

The zl command displays the default command string.

Example

The following example displays the default command string:

zl

The resulting output is similar to the following:

"R "

## zs WDEB386 command

zs "string"

The zs command makes it possible for you to change the default command string.

| Parameter | Description                                                                                                                                                                           |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| string    | Specifies the new default command string. The string must be enclosed in single or double quotation marks. You must separate the debugger commands within the string with semicolons. |

### Example

The following example changes the current default command string to an r (Display Register) command followed by a c (Compare Memory) command:

```
zs "r;c100 L 100 300"
```

The following example begins execution whenever an int 3 instruction is executed in your test application. This example executes a g (Go) command every time an int 3 instruction is executed.

```
zs "j (by cs:ip) == cc 'g' "
```

You can use zs as follows to set up a watchpoint:

```
zs "j (wo 40:1234) == 0eeed;t"
```

This command traces until the word at 40:1234 is not equal to 0EEED. This does not work if you are tracing through the mode switching code in MS-DOS or other sections of code that cannot be traced.

## CodeView for Windows

The Microsoft CodeView for Windows (CVW) debugger is a powerful, easy-to-use tool for the Microsoft Windows operating system. With CVW, you have the power to test the execution of your application and examine your data simultaneously. You can isolate problems quickly because you can display any combination of variables--global or local--while you interrupt or trace an application's execution.

CVW provides a variety of ways to analyze an application. You can use the debugger to examine source code, disassemble machine code, or examine a mixed display that shows you precisely which machine instructions correspond to each of your C-language statements. You can also monitor the occurrence of specific Windows messages.

CVW is similar to Microsoft CodeView (CV) version 3.0 for Microsoft®MS-DOS®. If you are familiar with CV for MS-DOS, see Differences Between CVW and CodeView for MS-DOS for a concise description of the unique features of CVW.

This topic serves as a complement to the CVW Help system. A significant portion of the CVW documentation is online. For information about using the CVW Help system, see Accessing Help.

Using CVW with a Single Monitor

CodeView for Windows version 3.07 allows you to debug Windows applications with a single monitor. See Using Codeview with a Single Monitor for more information.

The following topics describe how to set up and use CodeView for Windows:

Requirements for Using CVW

Comparing CVW with Other Microsoft Debuggers

Preparing to Run CVW

Starting a Debugging Session

Saving Session Information

Using Codeview with a Single Monitor

Working with the CVW Screen

Getting On-line Help in CVW

Displaying Program Data

Modifying Program Data

Controlling Program Execution

Handling Abnormal Termination of the Application

Ending a CVW Session

Advanced CVW Techniques

Customizing CVW with the TOOLS.INI File

Note: CVW supports the Microsoft Mouse or any fully compatible pointing device. This topic describes both mouse and keyboard procedures.



## Requirements for Using CVW

### Requirements for Using CodeView for Windows

Following are the system requirements for using CVW:

- Your system must have at least 384K of extended memory. For applications compiled with many symbols, 1 megabyte or more of extended memory is required.
- For 80386-based systems, the following required entry is automatically added to the [386enh] section of your SYSTEM.INI file when you install CVW:

```
device=windebug.386
```

- Your PATH environment variable must include the directory (or directories) containing CVW3.EXE, CVWIN.DLL, WINDEBUG.386, and CVW3.HLP.

## Using CVW with a Single Monitor

### Using CVW with a Single Monitor

It is possible to use CVW version 3.07 with a single monitor. For single-monitor debugging, you must have one of the following:

- A VGA display. CVW directly supports single-monitor debugging with a VGA display in both 386 enhanced and standard modes. No additional driver is needed.
- An EGA or other display with an 80386-based or 80486-based system running in 386 enhanced mode (you must use a VGA display in standard mode). With a non-VGA (or nonstandard VGA) display, you must install the VCV.386 driver. Place the driver in your Windows \SYSTEM directory and add the following entry to the [386enh] section of your Windows SYSTEM.INI file:

```
device=vcv.386
```

### Using CVW with a Secondary Monitor

You may find it more convenient to use a dual-monitor configuration. With the secondary monitor connected to your system, you can view CVW output and Windows output simultaneously. (CVW version 3.07 does not support a serial terminal.)

If you are using a secondary monochrome monitor for your CVW display, you need a monochrome adapter card and monochrome display monitor.

To set up a secondary monitor for debugging, do the following:

- 1 Install a secondary monochrome adapter card in a free slot in your computer, and connect the monochrome monitor to the port in the back.
- 2 Set the switches for the secondary display adapter to the appropriate settings, according to the display adapter and computer manufacturers' recommendations.

To use the secondary monochrome monitor, you must specify the /2 option on the command line when you start CVW.

If your system is an IBM Personal System/2, it must be configured with an IBM 8514/a display as the primary monitor and a VGA display as the secondary monitor. To use this configuration, specify the /8 (8514/a) option on the cvw command line when you choose the Run command from the File menu in Program Manager. If your VGA display is monochrome, you must also use the /b (black-and-white) option. The 8514/a display serves as the Windows screen and the VGA display as the debugging screen.

Do not attempt to run non-Windows applications or MS-DOS Shell while running CVW with the /8 option.

By default, the debugging screen operates in 50-line mode in this configuration. If you specify the /8 option, you can optionally specify the /25 or /43 option for 25- or 43-line mode, respectively, on the VGA debugging screen.

For more information about the command-line display options for CVW, see Display Options.

## Comparing CVW with Other Microsoft Debuggers

### Comparing CodeView for Windows with Other Microsoft Debuggers

If you have programmed in the Windows environment, you may have used the Microsoft Symbolic Debugger (SYMDEB) to debug Windows applications. You may also be familiar with CodeView (CV) for MS-DOS. This section describes the features and functions of CVW that are different from the features and functions of these other Microsoft debugging tools.

#### Differences Between CVW and SYMDEB

CVW has all the capabilities of SYMDEB and a number of features that SYMDEB does not provide. Following is a summary of the differences between SYMDEB and CVW:

| SYMDEB feature                                                          | CVW feature                                                                                                                                                                                  |
|-------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Debugs applications in real mode.                                       | Debugs applications in protected mode.                                                                                                                                                       |
| Examines only global (static) variables.                                | Examines both global and local variables.                                                                                                                                                    |
| Examines memory only when you specify simple memory addresses or symbol | .2Examines memory directly, but also uses the C-language expression evaluators to combine any variables with higher-level-language syntax. Provides only breakpoints to interrupt execution. |
| Does not set breakpoints or tracepoints on Windows messages.            | Sets breakpoints and tracepoints on Windows messages.                                                                                                                                        |
| Works through command line.                                             | Works through command line or menus.                                                                                                                                                         |

#### Differences Between CVW and CodeView for MS-DOS

With CVW, as with CV for MS-DOS, you can display and modify any variable, section of addressable memory, or processor register; monitor the path of execution; and precisely control where execution pauses. However, CV for MS-DOS and CVW differ in the following ways:

| CV feature                                                                 | CVW feature                                                                                                                                                                    |
|----------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Starts from the MS-DOS prompt.                                             | Starts from within Windows.                                                                                                                                                    |
| Repeats a search when you press ALT+/. Returns to MS-DOS upon termination. | Repeats a search when you press CTRL+R. Returns to Windows under normal termination conditions. An abnormal termination of CVW may cause the Windows session to be terminated. |

In addition to these differences, CVW includes the following unique features:

- The ability to track your application's segments and data as Windows moves their locations in memory. As items are moved, the debugger readjusts its symbol table accordingly.
- The (lh) and (gh) type casts, which you can use to dereference local and global handles of a memory object into near and far pointer addresses. For a more detailed description, see Dereferencing Memory Handles.
- Windows-specific commands. CVW has the following six new commands:

| Command                           | Action                                                                            |
|-----------------------------------|-----------------------------------------------------------------------------------|
| wdl (Windows Display Local Heap)  | Displays a list of the memory objects in the local heap.                          |
| wdg (Windows Display Global Heap) | Displays a list of the memory objects in the global heap.                         |
| wdm (Windows Display Modules)     | Displays a list of the application and library modules available to Windows.      |
| wwm (Windows Watch Message)       | Displays a Windows message or class of messages in the <u>CVW</u> Command window. |
| wbm (Windows Breakpoint Message)  | Sets a breakpoint on a Windows message or class of messages.                      |
| wka (Windows Kill Application)    | Terminates the task that is running. You should use this command with caution.    |

## Preparing to Run CVW

### Preparing Windows Applications for Debugging

If you want to use symbolic information and access source files with CVW, preparation depends on your compiler and linker.

Suppose, for example, that you were using Microsoft C Optimizing Compiler (CL), version 5.1 or later, and Microsoft Segmented Executable Linker (LINK). You would compile with the /Zi option to produce object files containing symbolic information and the /Od (disable optimization) option to ensure that code generated by the compiler would match the statements in the C-language source code. You would link with the /co option to produce an executable file containing symbolic information.

For further information about the settings you need to use, see the documentation that accompanied your compiler and linker.

### Setting Up the Debugging Version of Windows

You can run CVW with either the debugging or retail version of Windows. The debugging version performs error checking that is not available with the retail version.

For example, the debugging version of Windows checks whether a window handle passed to a Windows function is valid. When the debugging version of Windows detects such an error, it reports a fatal exit. If this happens while you are running CVW, the fatal exit is reported in the CVW Command window. For details about this error handling, see "Handling Abnormal Termination of the Application."

When you use the debugging version of Windows with CVW, the Windows core dynamic-link libraries (DLLs) provide debugging support. These DLLs (KRNL286.EXE, KRNL386.EXE, GDI.EXE, and USER.EXE) contain symbol information that makes it easier to determine the cause of an error. For example, if your application were to cause a general protection (GP) fault while running with the debugging version, Windows would display symbol information for the Windows code that was running when the GP fault was detected. If, instead, your application were running with the retail version of Windows, Windows would be able to display only CS:IP address values of the code that was being executed when the fault occurred.

CVW does not automatically use these Windows core DLL symbols. To provide CVW access to these symbols, you must specify one or more of the core DLLs either by using the /l command-line option or in response to the DLL prompt within CVW. If you are running CVW with Windows in standard mode, specify KRNL286.EXE. In 386 enhanced mode, specify KRNL386.EXE. For an explanation of how to load symbols from a DLL, see "Starting a Debugging Session for Dynamic-Link Libraries."

To install the debugging version of Windows, run the batch program N2D.BAT from your Windows system directory. This batch program replaces the nondebugging Windows core files with the debugging versions. (It copies both symbol files and executable files.) When the batch program has finished running, you start the debugging version of Windows by typing the win command. No special command-line options are required. To restore the nondebugging version of Windows, follow the same procedure using the batch program D2N.BAT.

## Starting a Debugging Session

As with Windows applications, you can start CVW in any of several ways. For a complete description of how to start Windows applications, see the Microsoft Windows User's Guide. To specify CVW options and parameters, you must choose the Run command from the File menu in Program Manager. For more information about CVW options, see "Command-Line Options."

You can run CVW to debug any of the following:

- A single application
- Multiple instances of an application
- Multiple applications
- DLLs

This section describes the methods you use to perform these tasks and summarizes the display options you can specify when you start CVW from the Run dialog box. This dialog box appears when you choose the Run command from the File menu in Program Manager.

### Display Options

You must specify your display selection on the command line when you start CVW. The following list describes the display options:

| Option                         | Display configuration                          |
|--------------------------------|------------------------------------------------|
| None                           | VGA; debugging on single monitor               |
| /v (VCV.386 must be installed) | Non-VGA; debugging on single monitor           |
| /2                             | Any; debugging on secondary monochrome monitor |
| /8                             | 8514/a; debugging on secondary VGA monitor     |

### Starting a Debugging Session for a Single Application

After you start CVW from Windows, CVW displays the Command Line dialog box. To start debugging a single application, do the following:

- 1 In the Command Line dialog box, type the name of the application. If you do not include an extension, CVW assumes the .EXE extension by default. You can also include any arguments that the application recognizes. Following is the syntax of the command to start debugging a single application:

```
app_name[.exe] [app_arguments]
```

- 2 Press ENTER, or choose the OK button.

CVW displays a dialog box with the following message:

```
Name any other DLL or executable with debug info.
```

- 3 Because you are debugging only one application and no DLLs, press ENTER or choose the OK button. CVW loads the application and displays on the debugging screen the source code for the application's WinMain function.
- 4 Set any breakpoints you want in the code.
- 5 To continue running the application, choose the <F5=Go> button on the status line or press the F5 key.

You can avoid startup dialog boxes and start CVW more quickly by specifying the application name as an argument on the command line, as follows:

- 1 From the Program Manager File menu, choose Run.
- 2 Type the application name and any application arguments on the command line. Following is the command syntax to start debugging a single application:

```
cvw [cvw_options] app_name[.exe] [app_arguments]
```

- 3 Press ENTER, or choose the OK button.

### Starting a Debugging Session for Multiple Instances of an Application

Windows can run multiple instances of an application simultaneously, which can cause a problem for your application. For example, two instances of an application might interfere with each other, or one application might corrupt the data of the other.

To help you solve problems associated with running multiple instances of an application, CVW allows you to debug multiple instances of an application at the same time. You can determine which instance of an application you are looking at by examining the DS register at any breakpoint.

To debug multiple instances of an application, perform the following steps:

- 1 Start CVW as usual for your application.
- 2 Run one or more additional instances of your application by choosing Run from the Program Manager File menu.

Specifying your application name more than once when starting CVW does not have the effect of loading multiple instances of the application.

The breakpoints you set in your application apply to all instances of the application. To determine which instance of the application has the current focus in CVW, examine the DS register.

#### Starting a Debugging Session for Multiple Applications

You can debug two or more applications at the same time, such as a dynamic data exchange (DDE) client and server. However, when global symbols are shared by applications (such as the symbol name WINMAIN), CVW resolves symbol references to the first application named when you started CVW.

Perform the following steps to debug two applications at the same time:

- 1 Start CVW as usual for a single application.
- 2 Type the name of the second application when CVW displays a dialog box with the following message:

Name any other DLL or executable with debug info.

You must include the .EXE extension after the filename of the second application.

- 3 Set breakpoints in either or both applications, choosing Open Module from the CVW File menu to display the source code for the different modules.
- 4 Press F5 to continue running the first application.
- 5 From the Program Manager File menu, choose Run, type the application name and any application arguments, and press ENTER or choose the OK button to start execution of the second application.

An alternative way to load the symbols for a second application is to use the /l option on the command line when you start CVW, as follows:

```
cvw /l second.exe first.exe
```

The /l option and the name of the second application must precede the name of the first application on the command line in the Run dialog box. You can repeat the /l option for each application to be included in the debugging session. Once CVW starts, choose the Run command from the Program Manager File menu to start the second application.

#### Starting a Debugging Session for Dynamic-Link Libraries

You can debug one or more DLLs while you are debugging an application. However, no distinction is made between global symbols shared by the applications and any DLLs.

Perform the following steps to debug a DLL at the same time as an application:

- 1 Start CVW as usual for the application.
- 2 Type the name of the DLL when CVW displays a dialog box with the following message:

Name any other DLL or executable with debug info.

CVW assumes the .DLL extension if you do not supply an extension with the filename. If your DLL has another extension (such as .DRV), you must specify it explicitly.

- 3 From the File menu, choose Open Module to display the source code for the different modules. Set breakpoints in either the application or the DLL.
- 4 Press F5 to continue running the application.

Alternatively, you can use the /l option to specify the DLL on the command line in the Run dialog box, as follows:

```
cvw /l appdll appname.exe
```

The /l option and the name of the DLL must precede the name of the first application on the command line. You can repeat the /l option for each DLL to be included in the debugging session. The .DLL extension is the default extension for the /l option.

CVW allows you to debug the LibEntry initialization routine of a DLL. If your application implicitly loads the library, a special technique is required to debug the LibEntry routine. An application implicitly loads a DLL if the library routines are imported in the application's module-definition (.DEF) file or if your application imports library routines through an import library when you link the application. An application explicitly loads a DLL by calling the LoadLibrary function.

If you type in the Command Line dialog box the name of an application that implicitly loads a DLL, CVW automatically loads the DLL and executes the DLL's LibEntry routine when CVW loads the application. In this case, you have no opportunity to debug the LibEntry routine. To avoid this problem, perform the following steps:

- 1 Instead of typing the name of your application in the Command Line dialog box, type the name of a dummy application that does not implicitly load the library.
- 2 Type the name of your DLL, being sure to include the extension if it is not .DLL, when the following message is displayed:

Name any other DLL or executable with debug info.

- 3 From the File menu, choose Open Module to display the source code for the library module containing the LibEntry routine. Set breakpoints in the LibEntry routine.
- 4 From the File menu, choose Open Module to display the source code for other library or application modules. Set breakpoints.
- 5 Press F5 to start running the dummy application.
- 6 Run the application that implicitly loads the DLL by choosing Run from the Program Manager File menu. CVW will resume control when the breakpoint in the LibEntry routine is encountered.

Alternatively, you can use a command line of the following form to specify the dummy application, your application, and the DLL:

```
cvw /l appdll dummyapp
```

After this command starts CVW, you need to perform steps 5 and 6 of the preceding procedure.

### Command-Line Options

Following is the command-line syntax to start CVW from the Run dialog box, which is displayed when you choose the Run command from the Program Manager File menu:

```
cvw [cvw_options] app_name[.exe] [app_arguments]
```

Parameters are not case-sensitive. Following are the command-line parameters:

|               |                                                                                                                                                                                                                                                                     |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cvw_options   | Specifies one or more options that modify how CVW runs. Options are not case-sensitive. Valid options are as follows:                                                                                                                                               |
| Option        | Purpose                                                                                                                                                                                                                                                             |
| /b            | Specifies a monochrome VGA display used as the secondary display with an 8514/a display. This option is valid only in conjunction with the /8 option.                                                                                                               |
| /c command    | Specifies one or more commands that CVW is to carry out when it loads the application specified by the app_name parameter. The group of commands must be enclosed in double quotation marks ("). Commands must be separated with semicolons (;).                    |
| /l dll_or_exe | Specifies the name of an application or DLL that has been compiled and linked with CVW symbols. CVW assumes the default filename extension .DLL if no extension is supplied. You can use the /l option more than once to specify multiple DLLs or executable files. |
| /m            | Disables the use of the mouse on the debugging screen. You should use this option when you set breakpoints in code that is responsive to mouse movements on the Windows application screen.                                                                         |
| /tsf          | Inverts save-state-file status for the current session.                                                                                                                                                                                                             |

|                |     |                                                                                                                                                                                                                            |
|----------------|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | /v  | Allows single-monitor debugging on a non-VGA display.                                                                                                                                                                      |
|                | /2  | Allows <u>CVW</u> to use a secondary monochrome monitor for debugger output while displaying Windows output on your primary monitor.                                                                                       |
|                | /8  | Allows <u>CVW</u> to use an 8514/a display as the Windows display and a <u>VGA</u> display for debugger output.                                                                                                            |
|                | /25 | Specifies 25-line mode for the secondary VGA display. This option is valid only in conjunction with the /8 option.                                                                                                         |
|                | /43 | Specifies 43-line mode for the secondary VGA display. This option is valid only in conjunction with the /8 option.                                                                                                         |
|                | /50 | Specifies 50-line mode for the secondary VGA display. This option is valid only in conjunction with the /8 option. The /50 option is not required, because 50-line mode is the default for the dual-monitor configuration. |
| app_name[.exe] |     | Specifies the location and name of the application for which <u>CVW</u> is to load symbols and issue an initial breakpoint. The .EXE extension is optional.                                                                |
| app_arguments  |     | Specifies one or more arguments recognized by the application that <u>CVW</u> loads.                                                                                                                                       |



## Saving Session Information

After your session, CVW stores session information in a file called CURRENT.STS, which is located in the directory pointed to by the INIT environment variable or in the current directory. If this file does not already exist, CVW automatically creates it. Session information includes the following:

- CVW display windows that were opened
- Breakpoint locations

CVW saves this information, which becomes the default session information the next time you run a CVW session for that application.

By default, this feature is enabled. You can disable this feature by placing the following entry in your TOOLS.INI file:

```
[cvw]
StateFileRead: n
```

The /tsf option temporarily inverts this setting when you run CVW. That is, if TOOLS.INI disables this feature, running CVW with the /tsf option saves session information for that session only.

If your Windows session abnormally terminates while CVW is running, the CURRENT.STS file may be corrupted. This may cause CVW to fail when it first tries to execute the application you are debugging. If this happens, delete the CURRENT.STS file before attempting to run CVW again.

Note: Microsoft Programmer's WorkBench (PWB) version 2.0 modifies the CURRENT.STS file. Once PWB has modified this file, CVW cannot read the command settings.

## Working with the CVW Screen

### Working with the CodeView for Windows Screen

When you start CVW, the CVW menu bar and three display windows--the Local window, the Source window, and the Command window--appear.

### Using CVW Display Windows

CVW divides the screen into logically separate sections called display windows, so that a large amount of information can be displayed in an organized and easy-to-read presentation. Each CVW display window is a distinct area on your monitor that operates independently of the other display windows. The name of each display window appears in the window's title bar. The following list describes the eight types of CVW display windows:

| CVW display window | Purpose                                                                                                                                                    |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Source window      | Displays the source code. You can open a second source window to view a header file, another source file, or the same source file at a different location. |
| Command window     | Accepts debugging commands.                                                                                                                                |
| Watch window       | Displays the current values of selected variables.                                                                                                         |
| Local window       | Lists the values of all variables local to the current function or block.                                                                                  |
| Memory window      | Shows the contents of memory. You can open a second Memory window to view a different section of memory.                                                   |
| Reg window         | Displays the contents of the microprocessor's registers, including flags.                                                                                  |
| 8087 window        | Displays the registers of the coprocessor or its software emulator.                                                                                        |
| Help window        | Displays the Help options or any Help information that you request.                                                                                        |

### Opening Display Windows

Following are the two ways to open CVW display windows:

- Choose a window from the View menu. (Note that you can open two Source windows and two Memory windows.)
- Perform an operation that automatically opens a window if it is not already open. For example, selecting a Watch variable automatically opens the Watch window.

CVW continually and automatically updates the contents of all its display windows.

### Selecting Display Windows

To select a window, click anywhere in it. You can also press F6 or SHIFT+F6 to move the focus from one window to the next.

The selected window is called the active window and is marked in three ways:

- The window's name is displayed in reverse video.
- The cursor appears in the window.
- Vertical and horizontal scroll bars appear in the window.

Typing commands in the Source window causes CVW to temporarily shift its focus to the Command window. Whatever you type is appended to the last line in the Command window. If the Command window is closed, CVW beeps in response to your input and ignores the input.

### Adjusting Display Windows

CVW display windows often contain more information than they can display on the screen. Although you cannot change the relative positions of the display windows, you can manipulate a selected window by using the mouse, as follows:

- To scroll through the information in the window, use the vertical or horizontal scroll bar.
- To maximize a window so that it fills the screen, click the Maximize arrow at the right end of the window's top border. To restore the window to its previous size and position, click the Maximize arrow again.
- To change the size of a window:
  - 1 Position the cursor anywhere on the border between two windows.
  - 2 Press and hold down the left mouse button.  
Two double-headed arrows appear on the line.
  - 3 Drag the mouse to enlarge or reduce the window.

- To close a window, click the Close box at the left end of the top border.  
 The adjacent windows automatically expand to recover the empty space.

You can also use the following keyboard commands:

| Keyboard command     | Description                                         |
|----------------------|-----------------------------------------------------|
| PAGE UP or PAGE DOWN | Scrolls through the text vertically.                |
| CTRL+F10             | Maximizes a selected display window.                |
| CTRL+F8              | Enables the arrow keys to resize the active window. |
| CTRL+F4              | Removes a selected display window.                  |

You can also choose the Maximize, Size, and Close commands from the View menu to manipulate a selected display window.

The different CVW display windows can help you to conduct a variety of debugging activities simultaneously. These activities are initiated and controlled with CVW debugging commands, which you can type on the command line when you start CVW or choose from CVW menus.

### Using the Menu Bar

In addition to display windows, the CVW screen includes a menu bar, which contains the following menus. For a more detailed description of CVW menus and commands, see CVW Help.

| Menu             | Contents                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|-------------|-------------|-------------------------------------------------------------------------------------------------------------------|---------------|----------------------------------------------------------------------------------------------------------------------------|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|------------------------------------------------------|-------|------------------------------------------------------|---------|--------------------------------------------------------|------|-----------------------------------------------------|----------|---------------------------------------------------------|------|-----------------------------------------------------|-------|---------------------------|
| File             | This menu contains the following commands:<br><table border="1"> <thead> <tr> <th>Command</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Open Source</td> <td>Opens any text file, and reads it into the active Source window.</td> </tr> <tr> <td>Open Module</td> <td>Opens the source file of any module for which CVW information has been loaded, and reads it into the active Source window.</td> </tr> <tr> <td>Exit</td> <td>Ends your CVW session, and returns you to Windows.</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Command | Description | Open Source | Opens any text file, and reads it into the active Source window.                                                  | Open Module   | Opens the source file of any module for which CVW information has been loaded, and reads it into the active Source window. | Exit             | Ends your CVW session, and returns you to Windows.                                                                                                                               |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Command          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Open Source      | Opens any text file, and reads it into the active Source window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Open Module      | Opens the source file of any module for which CVW information has been loaded, and reads it into the active Source window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Exit             | Ends your CVW session, and returns you to Windows.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Edit             | This menu contains the following commands:<br><table border="1"> <thead> <tr> <th>Command</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Undo</td> <td>Retracts the most recent edit, and restores the current line to its previous condition.</td> </tr> <tr> <td>Copy</td> <td>Copies selected text to the paste buffer.</td> </tr> <tr> <td>Paste</td> <td>Inserts text from the paste buffer into the active window at the present cursor location, if that location is valid (for example, text cannot be pasted into the Source window).</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Command | Description | Undo        | Retracts the most recent edit, and restores the current line to its previous condition.                           | Copy          | Copies selected text to the paste buffer.                                                                                  | Paste            | Inserts text from the paste buffer into the active window at the present cursor location, if that location is valid (for example, text cannot be pasted into the Source window). |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Command          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Undo             | Retracts the most recent edit, and restores the current line to its previous condition.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Copy             | Copies selected text to the paste buffer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Paste            | Inserts text from the paste buffer into the active window at the present cursor location, if that location is valid (for example, text cannot be pasted into the Source window).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| View             | This menu contains the following commands:<br><table border="1"> <thead> <tr> <th>Command</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Source</td> <td>Opens a new Source window.</td> </tr> <tr> <td>Memory</td> <td>Opens a new Memory window.</td> </tr> <tr> <td>Register</td> <td>Acts as a switch to open and close the Reg window.</td> </tr> <tr> <td>8087</td> <td>Acts as a switch to open and close the 8087 window.</td> </tr> <tr> <td>Local</td> <td>Acts as a switch to open and close the Local window.</td> </tr> <tr> <td>Watch</td> <td>Acts as a switch to open and close the Watch window.</td> </tr> <tr> <td>Command</td> <td>Acts as a switch to open and close the Command window.</td> </tr> <tr> <td>Help</td> <td>Acts as a switch to open and close the Help window.</td> </tr> <tr> <td>Maximize</td> <td>Enlarges the active window so that it fills the screen.</td> </tr> <tr> <td>Size</td> <td>Enables the arrow keys to resize the active window.</td> </tr> <tr> <td>Close</td> <td>Closes the active window.</td> </tr> </tbody> </table> | Command | Description | Source      | Opens a new Source window.                                                                                        | Memory        | Opens a new Memory window.                                                                                                 | Register         | Acts as a switch to open and close the Reg window.                                                                                                                               | 8087           | Acts as a switch to open and close the 8087 window.                                                                                                                                    | Local | Acts as a switch to open and close the Local window. | Watch | Acts as a switch to open and close the Watch window. | Command | Acts as a switch to open and close the Command window. | Help | Acts as a switch to open and close the Help window. | Maximize | Enlarges the active window so that it fills the screen. | Size | Enables the arrow keys to resize the active window. | Close | Closes the active window. |
| Command          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Source           | Opens a new Source window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Memory           | Opens a new Memory window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Register         | Acts as a switch to open and close the Reg window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| 8087             | Acts as a switch to open and close the 8087 window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Local            | Acts as a switch to open and close the Local window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Watch            | Acts as a switch to open and close the Watch window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Command          | Acts as a switch to open and close the Command window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Help             | Acts as a switch to open and close the Help window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Maximize         | Enlarges the active window so that it fills the screen.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Size             | Enables the arrow keys to resize the active window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Close            | Closes the active window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Search           | This menu contains the following commands:<br><table border="1"> <thead> <tr> <th>Command</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Find</td> <td>Searches for the next occurrence of a text string or a regular expression that you supply in the Find dialog box.</td> </tr> <tr> <td>Selected Text</td> <td>Searches for the next occurrence of a string of selected text.</td> </tr> <tr> <td>Repeat Last Find</td> <td>Searches for the next occurrence of the string or regular expression specified in the previous Find dialog box.</td> </tr> <tr> <td>Label/Function</td> <td>Searches for a label definition or function in the active Source window; if one is found, moves the input focus to the found label definition or function in the active Source window.</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                  | Command | Description | Find        | Searches for the next occurrence of a text string or a regular expression that you supply in the Find dialog box. | Selected Text | Searches for the next occurrence of a string of selected text.                                                             | Repeat Last Find | Searches for the next occurrence of the string or regular expression specified in the previous Find dialog box.                                                                  | Label/Function | Searches for a label definition or function in the active Source window; if one is found, moves the input focus to the found label definition or function in the active Source window. |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Command          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Find             | Searches for the next occurrence of a text string or a regular expression that you supply in the Find dialog box.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Selected Text    | Searches for the next occurrence of a string of selected text.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Repeat Last Find | Searches for the next occurrence of the string or regular expression specified in the previous Find dialog box.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |
| Label/Function   | Searches for a label definition or function in the active Source window; if one is found, moves the input focus to the found label definition or function in the active Source window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |         |             |             |                                                                                                                   |               |                                                                                                                            |                  |                                                                                                                                                                                  |                |                                                                                                                                                                                        |       |                                                      |       |                                                      |         |                                                        |      |                                                     |          |                                                         |      |                                                     |       |                           |

| Run              | This menu contains the following command:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|-------------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|---------------------------------------------------------------|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|--------------------------------------------------------------------------------------------------------|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  | <table border="0"> <thead> <tr> <th>Command</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Animate</td> <td>Continues running an application while displaying the execution path in the Source window. This type of display is called an animated trace display.</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Command | Description | Animate       | Continues running an application while displaying the execution path in the Source window. This type of display is called an animated trace display. |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Command          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Animate          | Continues running an application while displaying the execution path in the Source window. This type of display is called an animated trace display.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Watch            | This menu contains the following commands:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
|                  | <table border="0"> <thead> <tr> <th>Command</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Add Watch</td> <td>Adds an expression to the Watch window.</td> </tr> <tr> <td>Delete Watch</td> <td>Deletes an expression from the Watch window.</td> </tr> <tr> <td>Set Breakpoint</td> <td>Specifies where to interrupt execution of an application. You can set breakpoints on lines of source code, variables, expressions, and Windows messages.</td> </tr> <tr> <td>Edit Breakpoints</td> <td>Performs editing functions on breakpoints; they can be added, removed, modified, enabled, or disabled.</td> </tr> <tr> <td>Quick Watch</td> <td>Selects one expression for the Quick Watch dialog box. For a description of the Quick Watch window, see Section 4.9.4, "Using the Quick Watch Command."</td> </tr> </tbody> </table> | Command | Description | Add Watch     | Adds an expression to the Watch window.                                                                                                              | Delete Watch  | Deletes an expression from the Watch window.                  | Set Breakpoint | Specifies where to interrupt execution of an application. You can set breakpoints on lines of source code, variables, expressions, and Windows messages. | Edit Breakpoints | Performs editing functions on breakpoints; they can be added, removed, modified, enabled, or disabled. | Quick Watch      | Selects one expression for the Quick Watch dialog box. For a description of the Quick Watch window, see Section 4.9.4, "Using the Quick Watch Command." |
| Command          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Add Watch        | Adds an expression to the Watch window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Delete Watch     | Deletes an expression from the Watch window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Set Breakpoint   | Specifies where to interrupt execution of an application. You can set breakpoints on lines of source code, variables, expressions, and Windows messages.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Edit Breakpoints | Performs editing functions on breakpoints; they can be added, removed, modified, enabled, or disabled.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Quick Watch      | Selects one expression for the Quick Watch dialog box. For a description of the Quick Watch window, see Section 4.9.4, "Using the Quick Watch Command."                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Options          | This menu contains the following commands:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
|                  | <table border="0"> <thead> <tr> <th>Command</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Source Window</td> <td>Sets the display characteristics of the active Source window.</td> </tr> <tr> <td>Memory Window</td> <td>Sets the display characteristics of the active Memory window.</td> </tr> <tr> <td>Trace Speed</td> <td>Sets the speed of tracing and execution of an application.</td> </tr> <tr> <td>Case Sensitivity</td> <td>Turns case sensitivity on or off.</td> </tr> <tr> <td>386 Instructions</td> <td>Reads all 80386 instructions as 32-bit values when this command is checked; otherwise, reads all instructions as 16-bit values.</td> </tr> </tbody> </table>                                                                                                                                              | Command | Description | Source Window | Sets the display characteristics of the active Source window.                                                                                        | Memory Window | Sets the display characteristics of the active Memory window. | Trace Speed    | Sets the speed of tracing and execution of an application.                                                                                               | Case Sensitivity | Turns case sensitivity on or off.                                                                      | 386 Instructions | Reads all 80386 instructions as 32-bit values when this command is checked; otherwise, reads all instructions as 16-bit values.                         |
| Command          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Source Window    | Sets the display characteristics of the active Source window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Memory Window    | Sets the display characteristics of the active Memory window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Trace Speed      | Sets the speed of tracing and execution of an application.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Case Sensitivity | Turns case sensitivity on or off.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| 386 Instructions | Reads all 80386 instructions as 32-bit values when this command is checked; otherwise, reads all instructions as 16-bit values.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Calls            | The contents and size of this menu change as your application runs. The Calls menu shows the currently executing routine and the trail of routines from which it was called. Your application must execute at least the beginning of the <u>WinMain</u> function before CVW will display the current routine. When you select one of the <u>lines</u> in the Calls menu, CVW displays the source code corresponding to the calling location in the active source window.                                                                                                                                                                                                                                                                                                                                                                        |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |
| Help             | This menu can be used to access Help.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |         |             |               |                                                                                                                                                      |               |                                                               |                |                                                                                                                                                          |                  |                                                                                                        |                  |                                                                                                                                                         |

## Getting On-line Help in CVW

### Accessing Help

CVW Help contains detailed information and examples not found in this topic. You can access Help by choosing a command from the Help menu described in the preceding section or by selecting an item on your screen and pressing F1. Help is available on such items as commands, menus, dialog boxes, and error messages.

## Displaying Program Data

### Displaying Application Data

CVW offers a variety of ways to display variables, processor registers, and memory. You can also modify the values of any of these items as the application runs. This section describes how to display the following:

- Variables in the Watch window
- Expressions in the Watch window
- Arrays and structures in the Watch window
- A single expression in the Quick Watch dialog box
- Windows messages in the Command window
- Memory in the Memory window
- Contents of registers in the Reg window

### Displaying Variables

You can use the Watch window to monitor the value of a given variable throughout the execution of your application. For example, do, for, and while loops can cause problems when they don't terminate correctly. By displaying loop variables in the Watch window, you can determine whether a loop variable achieves its proper value.

To add a variable to the Watch window, perform the following steps:

- 1 In the Source window, use the mouse or the arrow keys to position the cursor on the name of the variable you want to watch.
- 2 From the Watch menu, choose Add Watch, or press CTRL+W.  
An Add Watch dialog box appears with the selected variable's name displayed in the Expression field.
- 3 Choose the OK button or press ENTER to add the variable to the Watch window.  
If you want to add a variable other than the one shown in the dialog box, type its name over the one displayed and press ENTER.

Adding a Watch variable opens the Watch window automatically if it is not already open. The Watch window appears at the top of the screen.

When you add a local variable, the following message may be displayed:

#### Watch Expression Not in Context

This message appears when execution has not yet reached the C-language function that defines the local variable. Global variables (those declared outside C-language functions) never cause CVW to display this message; you can watch them from anywhere in the application.

If any two or more applications or DLLs you are debugging contain global variables with the same name, CVW displays the variable of only the first application or DLL containing that variable name.

For example, if you are debugging App1 and App2, which both contain a global variable named hInst, CVW always displays the value of hInst in App1--even if CVW stopped at a breakpoint in App2.

The Watch window can display as many variables as you like; the quantity is limited only by available memory. You can scroll through information in the Watch window to view other variables. CVW automatically updates all watched variables as the application runs, including those not currently visible.

To remove a variable from the Watch window, do the following:

- 1 From the Watch menu, choose Delete Watch.
- 2 Scroll through information in the Delete Watch dialog box, and select the variable you want to remove.

Alternatively, you can position the cursor on any line in the Watch window and press CTRL+Y to delete the line.

### Displaying Expressions

You may have noticed that the Add Watch dialog box prompts for an expression, not simply a variable name. You can add any valid combination of variables, constants, or operators as an expression for CVW to evaluate and display in the Watch window.

The advantage of evaluating expressions is that you can reduce several variables to a single value, which may be easier to interpret than the components that make it up. For example, imagine a for loop in

which the ratio between two variables, var1 and var2, should remain constant. You suspect that one of these variables sometimes has the wrong value. To see when the quotient changes, without having to mentally divide two numbers, you can specify the following expression for display in the Watch window:

```
(var1 / var2)
```

You can also display Boolean expressions. For example, if the variable var is never supposed to be greater than 100 or less than 25, the following expression evaluates to 1 (TRUE) when var exceeds its limits:

```
(var < 25 || var > 100)
```

### Displaying Arrays and Structures

An application variable is usually a scalar quantity (a single character, integer, or floating-point value). The variable appears in the Watch window with the variable name to the left, followed by an equal sign (=) and the current value.

The Watch window provides a different way to display aggregate data items, such as arrays and structures. Arrays and structures contain multiple values that can be arranged in one or more layers. You can control how these variables appear in the Watch window--whether all, part, or none of their internal structure is displayed.

For example, the array WordHolder initially appears in the Watch window in the following form:

```
+WordHolder[] = [...]
```

The brackets indicate that this variable contains more than one element. The plus sign (+) indicates that the variable has more elements than are displayed on the screen. You can expand the variable to display any or all of its components; this technique is called dereferencing.

To dereference (expand) the array, you can double-click anywhere on the displayed line or you can position the cursor on the line and press ENTER. For example, if WordHolder is a six-character array containing the word Basic, the Watch window display changes to the following:

```
-WordHolder[]  
  [0] = 66 'B'  
  [1] = 97 'a'  
  [2] = 115 's'  
  [3] = 105 'i'  
  [4] = 99 'c'  
  [5] = 0 ''
```

Note that both the individual character values and their ASCII decimal equivalents are listed. The minus sign (-) indicates that no further expansion is possible. To contract the array, you can double-click its line again or you can position the cursor on the line and press ENTER.

### Displaying Character Arrays

If viewing a character array in this form is inconvenient, use either of the following methods to specify the watchpoint:

- Type the variable name, a comma (,), and the letter s, as shown in the following example:

```
WordHolder,s
```

CVW displays the contents of the array, as follows:

```
WordHolder,s[] = "Basic"
```

- Cast the variable's name to a character pointer, as shown in the following example:

```
(char *)WordHolder
```

CVW displays the address of the array and its contents, as follows:

```
(char *)WordHolder = 0x8C7:0x0010 "Basic"
```

### Displaying Multidimensional Arrays

You can display an array with more than one dimension. For example, imagine an integer array (5 by 5) named Matrix, whose diagonal elements are the numbers 1 through 5 and whose other elements are zero. Unexpanded, the array is displayed like this:

```
+Matrix[] = [...]
```

Double-click on the word Matrix (or position the cursor on that line and press ENTER) to change the display to the following:

```
-Matrix[]
+[0][] = [...]
+[1][] = [...]
+[2][] = [...]
+[3][] = [...]
+[4][] = [...]
```

The actual values of the elements are not shown yet. You have to descend one more level to see them. For example, to view the elements of the third row of the array, position the cursor anywhere on its subscript line (the +[2] line) and press ENTER. The following example shows the third row of the array dereferenced:

```
-Matrix[]
+[0][] = [...]
+[1][] = [...]
-[2][]
[0] = 0
[1] = 0
[2] = 3
[3] = 0
[4] = 0
+[3][] = [...]
+[4][] = [...]
```

Dereferencing the fifth row (+[4]) of the array produces this display:

```
-Matrix[]
+[0][] = [...]
+[1][] = [...]
-[2][]
[0] = 0
[1] = 0
[2] = 3
[3] = 0
[4] = 0
+[3][] = [...]
-[4][]
[0] = 0
[1] = 0
[2] = 0
[3] = 0
[4] = 5
```

Any element of an array or structure can be independently expanded or contracted; you need not display every element of the variable. If you want to view only one or two elements of a large array, specify the particular array or structure elements in the Expression field of the Add Watch dialog box.

You can dereference a pointer in the same way as an array or structure. The Watch window displays the pointer address, followed by all the elements of the variable to which the pointer currently refers. You can display multiple levels of indirection (that is, pointers referencing other pointers) simultaneously.

#### Displaying Dynamic Array Elements

An array may have dynamic elements that change as some other variable changes. Just as you can display a particular element of an array by selecting its subscript, you can also display a dynamic array element by specifying its variable subscript. For example, suppose that the loop variable *p* is a subscript for the array variable *Catalogprice*. The Watch window expression *Catalogprice[p]* displays only the array element currently specified by the variable *p*, not the entire array.

You can mix constant and variable subscripts. For example, the expression *BigArray[3][i]* displays only the element in the third row of the array to which the index variable *i* points.

#### Using the Quick Watch Command



Using the Quick Watch command is a convenient way to take a quick look at a variable or expression. Because the Quick Watch dialog box can display only one variable at a time, it's best to use the Watch window to view most variables.

Selecting the Quick Watch command from the Watch menu (or pressing SHIFT+F9) displays the Quick Watch dialog box. If the cursor is in the Source, Local, or Watch window, the variable at the current cursor position appears in the Quick Watch dialog box.

The Quick Watch display automatically expands arrays and structures to their first level. For example, an array with three dimensions expands to the first dimension. You can expand or contract an element just as you would in the Watch window; position the cursor on the appropriate line and press ENTER. If the array has more lines than the Quick Watch dialog box can display, you can view the rest of the array either by using the scroll bar or by pressing the DOWN ARROW or PAGE DOWN key.

To add a Quick Watch item to the Watch window, choose the Add Watch button. Arrays and structures appear in the Watch window expanded as they were displayed in the Quick Watch dialog box.

You can also display a Quick Watch dialog box for a variable by typing two question marks and the variable name in the Command window. For example, the following command shows the contents of the Index variable:

```
?? Index
```

### Tracing Windows Messages

You can trace occurrences of a Windows message or an entire class of Windows messages by using the wwm (Windows Watch Message) command. CVW displays the messages in the CVW Command window.

To trace a Windows message or message class, type the wwm command in the Command window. The syntax for the command is as follows:

```
wwm winproc msgname | msgclasses
```

The winproc parameter is the symbol name or address of an application's window procedure. The msgname parameter is the name of a Windows message, such as WM\_PAINT. The msgclasses parameter is a string of characters that identify one or more classes of messages to be traced. If msgclasses is not specified, CVW traces all message classes. The class, if specified, is consistent with those defined in Microsoft Windows Spy (SPY.EXE); they are as follows:

| Message class | Type of Windows message |
|---------------|-------------------------|
| c             | Clipboard               |
| d             | DDE                     |
| i             | Initialization          |
| m             | Mouse                   |
| n             | Input                   |
| s             | System                  |
| w             | Window management       |
| z             | Nonclient               |

For example, the following command traces all mouse and input messages sent to the MainWndProc procedure:

```
wwm MainWndProc mn
```

The following example illustrates how the CVW Command window displays a Windows message:

```
HWND:1c00 wParm:0000 lParm:000000 msg:000F WM_PAINT
```

### Displaying Memory

Selecting the Memory command from the View menu opens a Memory window. You can have two CVW Memory windows open at a time.

By default, memory is displayed as byte values in hexadecimal format, with 16 bytes per line. At the end of each line is a second display of the same memory in ASCII form. Values that correspond to printable ASCII characters (decimal values 32 through 127) are displayed in decimal format. Values outside that range are represented by periods (.).

Byte values are not always the most convenient way to view memory. If the area of memory you are examining contains character strings or floating-point values, you might prefer to view them in a directly readable form. The Memory Window command on the Options menu displays a dialog box with the display options in the following categories:

- ASCII characters
- Byte, word, or doubleword binary values
- Signed or unsigned integer decimal values
- Short (32-bit), long (64-bit), or 10-byte (80-bit) floating-point values

You can also cycle through these display formats directly by pressing SHIFT+F3.

If a section of memory cannot be displayed as a valid floating-point number, the value shown includes the characters NAN (not a number).

#### Displaying Local and Global Memory Objects

CVW is also useful for displaying global and local memory objects in their respective Windows heaps.

You can use the wdg (Windows Display Global Heap) command to display the entire heap of global memory objects in the Command window, or you can use the wdl (Windows Display Local Heap) command to display the entire heap of local memory objects in the Command window.

For the wdg command, you can specify a global handle to display a partial list of the global heap. The Command window displays the first five memory objects in the global heap, starting at the handle rather than at the beginning of the heap. The following example illustrates the wdg output format:

```

(1)      (2) (3)  (4) (5)      (6)
047E (0A7D) 00000020b MYAPP PRIV MOVEABLE DISCARDABLE

(7)
0A6D00000134b MYAPP DATA FIXED PGLOCKED=0001

(8)
0806 (0805) 00000600b PDB (0465)

(9)
FREE000000A0b

```

The following table describes the indicated fields:

| Field         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |             |         |               |                                                                                             |      |              |      |                                    |      |                                       |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|---------|---------------|---------------------------------------------------------------------------------------------|------|--------------|------|------------------------------------|------|---------------------------------------|
| 1             | The value of the handle of a global memory object. Global memory objects are displayed in the order in which Windows manages them, which is typically not in ascending handle order.                                                                                                                                                                                                                                                                                                |             |         |               |                                                                                             |      |              |      |                                    |      |                                       |
| 2             | A memory selector. This value is not displayed if the selector value is the same as the global handle, as is the case for DATA objects.                                                                                                                                                                                                                                                                                                                                             |             |         |               |                                                                                             |      |              |      |                                    |      |                                       |
| 3             | The length, in bytes, of the global memory object.                                                                                                                                                                                                                                                                                                                                                                                                                                  |             |         |               |                                                                                             |      |              |      |                                    |      |                                       |
| 4             | The name of the application or library module that allocated the object.                                                                                                                                                                                                                                                                                                                                                                                                            |             |         |               |                                                                                             |      |              |      |                                    |      |                                       |
| 5             | The type of global memory object, which can be the following: <table border="1" data-bbox="370 1402 1047 1564"> <thead> <tr> <th>Type</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>PRIV</td> <td>Application or DLL global data, or system object</td> </tr> <tr> <td>CODE</td> <td>Code segment</td> </tr> <tr> <td>DATA</td> <td>Data segment of application or DLL</td> </tr> <tr> <td>FREE</td> <td>Free memory object in the global heap</td> </tr> </tbody> </table> | Type        | Meaning | PRIV          | Application or DLL global data, or system object                                            | CODE | Code segment | DATA | Data segment of application or DLL | FREE | Free memory object in the global heap |
| Type          | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |             |         |               |                                                                                             |      |              |      |                                    |      |                                       |
| PRIV          | Application or DLL global data, or system object                                                                                                                                                                                                                                                                                                                                                                                                                                    |             |         |               |                                                                                             |      |              |      |                                    |      |                                       |
| CODE          | Code segment                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |             |         |               |                                                                                             |      |              |      |                                    |      |                                       |
| DATA          | Data segment of application or DLL                                                                                                                                                                                                                                                                                                                                                                                                                                                  |             |         |               |                                                                                             |      |              |      |                                    |      |                                       |
| FREE          | Free memory object in the global heap                                                                                                                                                                                                                                                                                                                                                                                                                                               |             |         |               |                                                                                             |      |              |      |                                    |      |                                       |
| 6             | One of the following memory allocation attributes: <p>MOVEABLE</p> <p>MOVEABLE DISCARDABLE</p> <p><u>FIXED</u></p>                                                                                                                                                                                                                                                                                                                                                                  |             |         |               |                                                                                             |      |              |      |                                    |      |                                       |
| 7             | One of the following dispositions if the object is movable: <table border="1" data-bbox="370 1780 1323 1866"> <thead> <tr> <th>Disposition</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>LOCKED=number</td> <td>Number of times the object has been locked with any of the Windows functions that lock data</td> </tr> </tbody> </table>                                                                                                                                    | Disposition | Meaning | LOCKED=number | Number of times the object has been locked with any of the Windows functions that lock data |      |              |      |                                    |      |                                       |
| Disposition   | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |             |         |               |                                                                                             |      |              |      |                                    |      |                                       |
| LOCKED=number | Number of times the object has been locked with any of the Windows functions that lock data                                                                                                                                                                                                                                                                                                                                                                                         |             |         |               |                                                                                             |      |              |      |                                    |      |                                       |

- PGLOCKED=number      Number of times Windows has locked the object in its linear address space
- 8      The handle of the application or library module that allocated the process descriptor block (PDB).
- 9      A free memory object, followed by the size of the free object, in bytes.

The following example shows sample output of the wdl (Windows Display Local Heap) command:

```
(1)      (2)      (3) (4)
190A: 000A    BUSY    (16DA)
```

The following table describes the indicated fields:

| Field | Description                                                     |
|-------|-----------------------------------------------------------------|
| 1     | The offset of the local memory object in the local data segment |
| 2     | The length of the object, in bytes                              |
| 3     | One of the following dispositions:<br>Disposition      Meaning  |
|       | BUSY              A currently allocated object                  |
|       | FREE              A free object in the local heap               |
| 4     | A local memory handle                                           |

#### Displaying Variables with a Live Expression

"Using the Quick Watch Command," explains how to display a specific array element by adding the appropriate expression to the Watch window. It is also possible to view a particular array element or structure element in the Memory window. This CVW display feature is called a live expression, because the displayed area of memory changes to reflect the value of a pointer or subscript. For example, if Buffer is an array and pBuf is a pointer to that array, then \*pBuf points to the array element currently referenced. A live expression displays the section of memory beginning with this element.

CVW displays live expressions in a Memory window. To create a live expression:

- 1 From the Options menu, choose Memory Window.
- 2 Select the Live Expression check box, and type the name of the element you want to view.  
For example, if pszMsg is a pointer to a null-terminated array of characters and you want to see what it currently points to, type the following:

```
*pszMsg
```

- 3 Choose the OK button, or press ENTER.

A new Memory window opens. The first memory location in the window is the first memory location of the live expression. The section of memory displayed changes to the section the pointer currently references.

You can use the Memory Window command on the Options menu to display the value of the live expression in a readable form. This is especially convenient when the live expression represents strings or floating-point values, which are difficult to interpret in hexadecimal form.

It is usually more convenient to view an item in the Watch window than as a live expression. However, you might find some items easier to view as live expressions. For example, you can examine what is currently at the top of the stack by specifying SS:SP as the live expression.

#### Dereferencing Memory Handles

In a Windows application, the LocalLock and GlobalLock functions are used to dereference memory handles into near or far pointers. In a debugging session, you may know the handle of the memory object, but might not know which near or far address it dereferences to, unless you are debugging in an area where the application has just completed a LocalLock or GlobalLock function call. To get the near and far pointer addresses for your local and global handles, use the (lh) and (gh) type casts. For example, you could use (lh) to dereference the array in the following code:

```
HANDLE hLocalMem;
PBYTE pbArray;

hLocalMem = LocalAlloc(LMEM_MOVEABLE, 100);
pbArray = (PBYTE)LocalLock(hLocalMem);
```

```
/* Use the array.*/
```

```
LocalUnlock(hLocalMem);
```

To properly display this array in CVW, you can use the following command:

```
dw (lh)hLocalMem
```

If you set a breakpoint immediately after the LocalLock function, you could find out where the local object was allocated in the application's data segment by looking at the value of the pbArray variable. To display the value of pbArray, use the following CVW command:

```
dw pbArray
```

Note that you cannot rely on the value of pbArray anywhere else in the application, because it may change or the memory object may move.

In the following example, the memory object lpszTest is a string:

```
HANDLE hGlobalMem;
```

```
LPSTR lpszTest;
```

```
hGlobalMem = GlobalAlloc(GMEM_MOVEABLE, 10L)
```

```
lpszTest = GlobalLock(hGlobalMem);
```

```
lstrcpy(lpszTest, "ABCDEF");
```

```
GlobalUnlock(hGlobalMem);
```

To display the contents of the string, you could use double type casting, as follows:

```
? *(char far*) (gh)lpszTest, s
```

The (gh) type cast returns a pointer to the far address of the global memory object.

#### Displaying the Contents of Registers

Selecting the Register command from the View menu (or pressing F2) opens a Reg window on the right side of the screen. The current values of the microprocessor's registers appear in this window.

At the bottom of the window are a group of mnemonics representing the processor flags. When your application first starts running, all values are shown in normal-intensity video. Any subsequent changes are marked in high-intensity video. For example, suppose the overflow flag is not set when the application starts. The corresponding mnemonic is NV, and it appears in normal-intensity video. If the overflow flag is subsequently set, the mnemonic changes to OV and appears in high-intensity video.

Selecting the 386 Instructions command from the Options menu displays the contents of the registers as 32-bit values. This command is valid only if your computer uses an 80386 processor. Selecting this command a second time changes the registers back to 16-bit values.

You can also display the registers of an 8087/80287/80387 coprocessor in a separate window by choosing the 8087 command from the View menu. If your application uses a coprocessor emulator, the emulated registers are displayed instead.

#### Displaying Windows Modules

The wdm (Windows Display Modules) command displays a list of all the DLL and task modules that Windows has loaded. For each module, the list shows the module handle, the type of module (DLL or task), the name of the module, and the path of the module.

## Modifying Program Data

### Modifying Application Data

You can easily change the values of variables, memory locations, or registers displayed in the Watch, Memory, Reg, or 8087 window. Simply position the cursor at the value you want to change, and type the appropriate value. If you change your mind, press `ALT+BACKSPACE` to undo the last change you made.

The Memory window displays the starting address of each line in segment:offset form. Altering the address automatically shifts the display to the corresponding section of memory. If that section is not used by your application, memory locations are displayed as double question marks (??). You cannot change memory that is displayed as question marks.

You can also change the values of memory locations by modifying the right side of the memory display, which shows memory values in ASCII form. For example, you can change a byte from decimal value 75 (ASCII value for uppercase K) to decimal value 85 (ASCII value for uppercase U). To do so, place the cursor over the letter K, which corresponds to the position where the memory value is 75, and type U.

To change a processor flag, you can click its mnemonic or you can position the cursor on a mnemonic and press any key (except `TAB` or `SPACEBAR`). Repeat these operations to restore the flag to its previous setting.

Although you can alter most items from the Watch window, sometimes it is useful to modify a register or memory directly. For example, if a function returns a value in the AX register, you can modify the AX register to change a returned value without executing the function.

**Warning:** You should be especially cautious when altering machine-level values. The effect of changing a register, flag, or memory location may vary from having no effect at all to causing the operating system to crash.

## Controlling Program Execution

### Controlling Execution of Your Application

This section describes how you can use CVW to control the execution of your application.

Following are the three possible forms of execution in CVW:

| Application execution | Description                                                                                                                                                            |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Continuous            | The application runs until either a previously specified breakpoint has been reached or the application terminates normally.                                           |
| Single-step           | The application pauses after each line of code has been executed.                                                                                                      |
| Animated              | The application pauses after each line of code has been executed, but execution continues after a short pause. The application continues to run until you press a key. |

### Continuous Execution

With continuous execution, you can quickly execute bug-free sections of code. To initiate continuous execution, either you can click the right mouse button on the line of code you want to debug or examine in more detail or you can position the cursor on this line and then press F7. Execution proceeds at full speed and pauses when it reaches the selected line.

You can also use a breakpoint to cause execution to pause at a specific line of code. CVW provides you with several types of breakpoints to control your application's execution. The sections that follow describe how to use breakpoints.

### Selecting Breakpoint Lines

By specifying one or more lines as breakpoints, you can skip over the parts of the application that you don't want to examine. Execution of the application proceeds at full speed up to the first breakpoint, at which execution is interrupted; pressing F5 causes execution to continue up to the next breakpoint; and so on. You can set as many breakpoints as you want, provided that you have available memory.

Following are several ways to set breakpoints:

- Double-click anywhere on the desired breakpoint line. The selected line is highlighted to show that it is a breakpoint. To remove the breakpoint, double-click on the line a second time.
- Position the cursor anywhere on the line at which you want execution to pause. Press F9 to select the line as a breakpoint and to highlight it. Press F9 a second time to remove the breakpoint and highlighting.
- Display the Set Breakpoint dialog box by choosing the Set Breakpoint command from the Watch menu. Select one of the breakpoint options that permits you to specify a line (location). The line on which the cursor rests is the default breakpoint line in the Location field. If this line is not the location you want, replace it by typing another line number in the Location field. When you type a new line number, make sure that you precede it with a period.
- Your application can call the Windows `DebugBreak` function to interrupt execution and return control to CVW. When your application calls the `DebugBreak` function, execution may stop within the `DebugBreak` code rather than in your application. You may have to single-step out of the `DebugBreak` code and back into your application.

A breakpoint line must contain executable code. You cannot select a blank line, a comment line, or a declaration line (such as a variable declaration or a preprocessor statement) as a breakpoint.

To set a breakpoint on a multiline statement, you must position the cursor on the last line of the statement. If you try to set a breakpoint on any other line of the statement, CVW does not accept it.

If your compiler optimizes your code, some lines of code may be repositioned or reorganized for more efficient execution. These changes can prevent CVW from recognizing the corresponding lines of source code as breakpoints. Therefore, it is a good idea to disable optimization during development. You can restore optimization once debugging is completed.

A breakpoint can also be set at a function or an explicit address. To set a breakpoint at a function, simply enter the name of the function in the Set Breakpoint dialog box. To set a breakpoint at an address, enter the address in CS:IP form.

If any of the applications or DLLs you are debugging share names for certain window procedures (such as `MainWndProc`), you can refer by name only to the procedure that is defined in the first application or DLL.

You can remove a breakpoint by choosing the Edit Breakpoints command from the Watch menu or by selecting the breakpoint in the Source window and pressing F9. When your application pauses at a

breakpoint, you can continue execution by pressing F5. You cannot remove a breakpoint set by an application calling the DebugBreak function.

#### Setting Breakpoint Values

Breakpoints are not limited to specific lines of code. CVW can also break execution when an expression changes value or reaches a particular value. Use one of the following methods to set a breakpoint value:

- To interrupt execution when an expression changes value, type the name of the expression in the Expression field of the Set Breakpoint dialog box.
- To interrupt execution when an expression reaches a particular value, use that value in the expression you type in the Expression field of the Set Breakpoint dialog box.

For example, if you want the application to pause when a variable named `looptest` equals 17, type the following in the Expression field:

```
looptest==17
```

The application pauses when this statement becomes true.

You can also use the Set Breakpoint dialog box to combine value breakpoints with line breakpoints so that execution stops at a specified line only if an expression has simultaneously changed value or reached a specified value.

For large variables (such as arrays and character strings), you can specify the number of bytes you want checked (up to 32K) in the Length field.

Note: When a breakpoint is tied to a variable, CVW must check the variable's value after each machine instruction is executed. This computational overhead slows execution greatly. For maximum speed when debugging, either tie value breakpoints to specific lines or set value breakpoints only after you have reached the section of code that needs to be debugged.

#### Setting Breakpoints on Windows Messages

You can also set a breakpoint on a Windows message or an entire class of Windows messages. By using this feature, you can track your application's response to user input and window-management messages.

To set a breakpoint on a Windows message or message class, type the `wbm` (Windows Breakpoint Message) command in the Watch window. The syntax for the command is:

```
wbm winproc msgname | msgclasses
```

The `winproc` parameter is the symbol name or address of an application's window procedure. The `msgname` parameter is the name of a Windows message, such as `WM_PAINT`. The `msgclasses` parameter is a string of characters that identify one or more classes of messages. If `msgclasses` is not specified, CVW traces all message classes. If it is specified, the classes are consistent with those defined in Microsoft Windows Spy (SPY.EXE); they are as follows:

| Message class | Type of Windows message |
|---------------|-------------------------|
| c             | Clipboard               |
| d             | DDE                     |
| i             | Initialization          |
| m             | Mouse                   |
| n             | Input                   |
| s             | System                  |
| w             | Window management       |
| z             | Nonclient               |

For example, if your application is failing to refresh the client area of a window, you might set a breakpoint on the `WM_PAINT` message so that you can watch your application's behavior as it processes the message. The following command interrupts execution whenever the application's `MainWndProc` procedure receives a `WM_PAINT` message:

```
wbm MainWndProc WM_PAINT
```

#### Using Breakpoints

This section shows how breakpoints can help you find the cause of a problem.

One of the most common bugs is a for loop that executes too many or too few times. If you set a breakpoint that encloses the loop statements, the application pauses after each iteration. You can then

monitor the loop variable or critical program variables in the Watch or Local window to find the error in loop processing.

You can specify that a breakpoint is to be ignored. To set the number of times a breakpoint is to be ignored before execution is interrupted, perform the following steps:

- 1 From the Watch menu, choose Set Breakpoint.
- 2 In the Pass Count field of the Set Breakpoint dialog box, type the decimal number.

For example, suppose your application repeatedly calls a function to create a binary tree. You suspect that something goes wrong approximately halfway through the process. You could mark the line that calls the function as the breakpoint, then specify how many times this line is to be executed before execution pauses. Running the application creates a representative (but unfinished) tree structure that can be examined from the Watch window. You can then continue your analysis by using single-step execution, which is described in the next section.

Another programming error is assignment of the wrong value to a variable. If you enter a variable in the Expression field of the Set Breakpoint dialog box, execution is interrupted every time the variable changes value.

Breakpoints make it possible for you to interrupt execution of an application so that you can assign new values to variables. For example, if a limit value is set by a variable, you can change the value to see if it affects the application's execution. Similarly, you can pass a variety of values to a switch statement to see if they are correctly processed. This ability to alter variables provides an especially convenient way to test new functions without having to write a stand-alone test application.

When your application reaches a breakpoint and you change a variable, you might want to watch each step be executed while you check the value of that variable. This technique is called single-stepping.

#### Single-Step Execution

When single-stepping, CVW pauses after each line of code is executed. If a line contains more than one executable statement, CVW executes all the statements on the line before pausing. The next line to be executed is displayed in reverse video. You can use either the Trace command or the Step command to single-step through an application.

To use Trace, press F8. Trace displays each step of every function for which CVW has symbolic information. Each line of the function is a separate step. If CVW does not have symbolic information for a function, the function runs in a single step.

To use Step, press F10. Step displays each step of the current function but does not step into function calls. Instead, the called function runs as a single step.

You can alternate between Trace and Step as you like. Which method you should use depends on whether you want to see what happens within a particular function.

Attempting to step or trace through Windows startup code while viewing assembly-language instructions causes unpredictable results. To step through your application while viewing assembly-language instructions, set a breakpoint at the WinMain function and begin stepping through the application only after the breakpoint has been reached.

Using the Trace command to step out of a window procedure causes CVW to step into Windows system code.

#### Animated Execution

To trace through the application continuously without having to press F8, choose the Animate command from the Run menu. The speed of execution is controlled by the Trace Speed command on the Options menu. You can interrupt animated execution at any time by pressing any key.

#### Jumping to a Particular Location

At times, you may wish to force the system to jump to a particular location in your application during execution. For example, you may want to avoid executing code that you know has bugs, or you may want to repeatedly execute a particularly troublesome portion of your application.

To jump to a specific location in your application, do the following:

- 1 From the Options menu, choose Source. Select the Mix Source and Assembly radio button and the Show Machine Code check box.
- 2 In the Source window, view the line of source code to which you want to jump.
- 3 Examine the code offset of the first machine instruction for the assembled statement.



- 4 To change the IP register to this code offset, type the rip (Register IP) command in the command window, supplying the value in hexadecimal format.

CVW highlights the line to which you have jumped.

**Warning:** Do not jump from one procedure to another. Jumping from one procedure to another disrupts the stack.

Assembled source code for a given statement may rely on memory values and registers set in previous instructions. If you cause execution to jump to a specific point in your application, values and registers may not be correctly set, particularly if optimization was not disabled during compiling.

#### Interrupting Your Application

There may be times when you want to interrupt your application immediately. You can force an immediate interruption of a CVW session by pressing CTRL+ALT+SYS RQ. You then have the opportunity to change debugging options; for example, you can add breakpoints and modify variables. To resume continuous execution, just press F5; to single-step, press F10.

You should take care when you interrupt the CVW session. For example, if you interrupt the session while Windows code or other system code is being executed, attempting to use the Step command or the Trace command could produce unpredictable results. When you interrupt the CVW session, it is usually safer to set breakpoints in your code and resume continuous execution than to use Step or Trace.

An infinite loop in your code presents a special problem. Again, because you should avoid using Step or Trace after interrupting your application, you should try to locate the loop by setting breakpoints in places you suspect are in the loop.

Whether or not you locate the infinite loop, you will have to terminate your application. The wka (Windows Kill Application) command terminates the task that is currently running. You should use the wka command only when your application is the one being executed.

If your application is currently executing a module that contains symbol information, the CVW Source window highlights the current instruction. However, if your application contains modules without symbolic information, it is more difficult to determine whether the assembly-language code displayed in the Source window belongs to your application or to another task.

In this case, use the wdg (Windows Display Global Heap) command, supplying the value in the CS register as the parameter. CVW displays a listing that indicates whether the code segment belongs to your application. If the code segment does belong to your application, you can use the wka command without affecting other tasks. The wka command does not perform all the cleanup tasks associated with the normal termination of a Windows application. For example, graphics device interface (GDI) objects created during the execution of the application but not destroyed before you terminated the application remain allocated in the systemwide global heap. This reduces the amount of memory available during your Windows session. Because of this, you should use the wka command to terminate the application only if you cannot terminate it normally.

The wka command simulates a fatal error in your application. Because of this, when you use the wka command, Windows displays an error message. After you close the message box, Windows may not release subsequent mouse input messages from the system queue until you press a key. If this happens, the cursor moves on the Windows screen, but Windows does not appear to respond to the mouse. After you press any key, Windows responds to all mouse events that occurred before you pressed the key.

## Handling Abnormal Termination of the Application

Your application can terminate abnormally in one of two ways while you are debugging it with CVW. It can cause a fatal exit, or it can cause a GP fault. In both cases, CVW regains control, giving you the opportunity to examine the state of the system when your application terminated. In particular, you can often determine the location in your application's code where the error occurred or which call caused the error. CVW makes it possible for you to view registers, display the global heap, display memory, and examine the source code.

Once you have determined where the error occurred, type the q (Quit) command in the Command window to terminate CVW. In most cases, control returns to Windows.

### Handling a Fatal Exit

If the abnormal termination was a fatal exit and the application was running with the retail version of Windows, CVW displays a fatal exit code and the CS:IP register contains an address in the Windows code itself. This small amount of information provides little to help you locate the last call that your application made before the error was detected.

If, however, your application was running with the debugging version of Windows, the CVW Command window displays a stack trace that is much more useful for finding the error in your source code.

After the stack trace appears in the CVW Command window, Windows prompts you with the following message:

Abort, Break, or Ignore?

To locate the cause of the error, press the B key. This allows CVW to regain control from Windows.

In most cases, the stack trace will have been scrolled past the top of the CVW Command window; but once CVW regains control, you can scroll the information in the window to examine the entire stack trace.

The following information appears at the top of the stack trace:

- A fatal exit number. For more information about Windows debugging messages, see Appendix C, "Windows Debugging Version."
- The CS:IP address, the name of the Windows function where the error was detected, or the name of the last Windows function called before the error was detected.

Following this information, additional Windows functions may be listed in the stack trace. Somewhere near the top of the stack trace, a CS:IP address is listed without a Windows function name. In most cases, this is the location in the source code of your application at which the call to a Windows function occurred, triggering the fatal exit.

To examine this location in your source code, open or switch to a Source window and use the v (View) command followed by the CS:IP address; be sure to precede both the segment and the offset with the hexadecimal prefix 0x. For example, if CVW indicates that the error occurred at 07DA:0543 in your application, type the following command:

```
v 0x07DA:0x0543
```

If the module at which the error occurred was compiled to produce object files containing symbolic information, the CVW Source window displays the location in your code at which the errant call to a Windows function occurred.

The first CS:IP address without a name in the stack trace may point to a location in your code without symbols. For example, the code may be in a DLL you didn't specify with the /l command-line option or when CVW prompted you for a DLL, or the address might be in a module that was not compiled to produce symbolic information. In such cases, CVW reports that no source code is available. If this happens, continue down the stack trace, using the v command to examine each unnamed CS:IP address. You are likely to find a location in a module that was compiled to produce symbolic information and to find this location made a call into one of your modules that was not compiled to produce symbolic information.

### Handling a General Protection Fault

When a general protection (GP) fault occurs, CVW displays a message in the Command window to notify you of the event. If the GP fault occurred at an instruction in one of your modules, CVW displays the corresponding source code if the module was compiled to produce symbolic information. You can obtain information about the chain of calls leading up to the GP fault by using the CVW Call menu. This menu displays a backtrace of calls in the form of a series of segments and offsets, starting at the most recent call.

If your application was running with the debugging version of Windows, the backtrace shows function names next to some of the segment:offset pairs. By examining the function names, you may be able to determine where in your code the error occurred.

## Ending a CVW Session

### Ending a Session

To terminate a CVW session, you can choose the Exit command from the File menu or type the q (Quit) command in the Command window.

You can also terminate your application without terminating CVW. While Windows is terminating the application, it notifies CVW. CVW then displays the following message:

```
Program terminated normally (0)
```

The value in parentheses is the return value of the WinMain function. This value is usually the wParam parameter of the WM\_QUIT message, which in turn is the value of the nExitCode parameter passed to the PostQuitMessage function.

If you were debugging more than one application or DLL, you can press F5 to continue the debugging session.

## Advanced CVW Techniques

### Advanced Techniques

Once you are comfortable displaying variables, changing variables, and controlling the execution of your application, you may want to experiment with the following advanced techniques:

- Using multiple Source windows
- Checking for undefined pointers
- Handling register variables
- Redirecting CVW input and output

### Using Multiple Source Windows

You can have two Source windows open at the same time. The windows can display two different sections of source code for the same application. They can both track CS:IP addresses, or one can display a high-level listing and one can display an assembly-language listing. You can move freely between the Source windows, executing a single line of source code or a single assembly-language instruction at a time.

### Checking for Undefined Pointers

Until a pointer has been explicitly assigned a value, its value is undefined. Its value can be completely random, or it can be some consistent value (such as 1) that does not point to a useful data address.

Accessing a value through an uninitialized pointer address can cause inexplicable or erratic application behavior, because the data is not being read from or written to the intended location. For example, suppose that var1 is mistakenly written to the address specified by an uninitialized pointer and that then var2 is written there. When var1 is read back, it does not have its original value, having been replaced by var2.

### Handling Register Variables

A register variable is stored in one of the microprocessor's registers, rather than in random-access memory (RAM). This speeds up access to the variable.

A conventional variable can become a register variable in either of the following ways:

- The variable is declared as a register variable. If a register is free, the compiler stores the variable there.
- The compiler stores a frequently used variable (such as a loop variable) in a register during optimization to speed up execution.

Register variables can cause problems during debugging. As with local variables, they are visible only within the function where they are defined. In addition, a register variable may not always be displayed with its current value.

Usually, it is a good idea to turn off all optimization and to avoid declaring register variables until the application has been fully debugged. Any side effects produced by optimization or register variables can then be easily isolated.

### Redirecting CodeView for Windows Input and Output

You can cause CVW to receive input from an input file and generate output to an output file. To redirect CVW input and output, you can use the /c option on a command line of the following form to start CVW:

```
cvw /c "<infile; t >outfile"
```

When you redirect input in this way, CVW carries out any commands in infile during startup. When CVW exhausts all commands in the input file, focus automatically shifts to the Command window.

When you redirect output, it is sent to both outfile and the Command window. You can use the t parameter before the right angle bracket (>) on the command line to send output to the Command window. You can also redirect output from the command line after CVW has started.

Redirection is a useful way to automate CVW startup. Although redirection makes it possible for you to keep a viewable record of command-line input and output, you cannot record mouse operations. Some applications--particularly interactive ones--may need modification to allow for redirection of input to the application itself.

## Customizing CVW with the TOOLS.INI File

### Modifying the TOOLS.INI File

To customize the behavior and user interface of CVW, modify the [cvw] section of your TOOLS.INI file. The TOOLS.INI file is an ASCII text file. You should place it in a directory pointed to by the INIT environment variable. (If you do not use the INIT environment variable, CVW looks for TOOLS.INI only in the CVW source directory.)

Most TOOLS.INI customizations control screen colors, but you can also specify startup commands or the name of the file that receives CVW output. The Help system contains complete information about all the TOOLS.INI entries for CVW.

## Monitoring DDE Transactions: DDESPY

DDESpy is a typical DDE monitoring application. Because DDE is a cooperative activity, DDE monitoring applications must follow certain guidelines for your Windows system to operate properly while they are in use. In particular, DDE monitoring applications should not perform DDE server or client communications--problems may arise when the monitoring application intercepts its own communications.

The following topics describe how to set up and use DDESPY:

Output Options  
Monitor Options  
Tracking Options

## DDESPY Output Options

### The Output Menu

DDESpy can display DDE information in a window or on your debugging terminal or can save the displayed information in a file for later use.

You use the Output menu to select where DDESpy is to send output. If you choose the File command, you must specify the name of an output file. After you have chosen the File command once, DDESpy prompts you for an output filename every time you restart the application.

From the Output menu, you can choose the Clear Screen command to clear the display window. You can choose the Mark command to add text to the display as a marker--for example, before a DDE event to make it easier to find the event in the output file.



## DDESPY Monitor Options

### The Monitor Menu

You use the Monitor menu to specify one or more types of DDE information that DDESpy is to display. The following information can be displayed:

String Handle Data

Sent DDE Messages

Posted DDE Messages

Callbacks

Errors

The Dynamic Data Exchange Management Library (DDEML) passes information by using shared memory. The contents of the shared memory depend on the type of DDE transaction. Several structures have been defined to allow applications using DDE to access the information in the shared memory. DDESpy displays the contents of the appropriate structure for the DDE activity being monitored.

## Monitoring String Handle Data

### Monitoring String-Handle Data

The DDEML uses the MONHSZSTRUCT structure to pass string-handle data. DDESpy displays the following information from this structure:

- Task (application instance)
- Time, in milliseconds, since you started Windows
- Activity type (create, destroy, or increment)
- String handle
- String contents

The following example shows a typical DDESpy display of string-handle data:

```
Task:0x94f, Time:519700, String Handle Created: c4a4(this is a test)
Task:0x94f, Time:526126, String Handle Created: c4aa(another test)
```

## Monitoring Sent and Posted DDE Messages

### Monitoring Sent or Posted DDE Messages

The DDEML uses the `MONMSGSTRUCT` structure to send and post DDE messages. DDESPY displays the following information from this structure:

- Task
- Time
- Handle of receiving window
- Transaction type (sent or posted)
- Message type
- Handle of sending application
- Other message-specific information

The following example shows a typical DDESpy display of DDE message activity:

```
Task:0x8df Time:642402 hwndTo=0x38dc Message(Sent)=Initiate:
  hwndFrom=9224, App=0xc35d("Server")
  Topic=*
Task:0x94f Time:642457 hwndTo=0x2408 Message(Sent)=Ack:
  hwndFrom=9396, App=0xc35d("Server")status=c35d(fAck fBusy )
  Topic=Item=0xc361("System")
```

## Monitoring Callbacks

### Monitoring Callbacks

The DDEML uses the `MONCBSTRUCT` structure to pass information to application callback functions. DDESpy displays the following information from this structure:

- Task
- Time
- Transaction type
- Exchanged-data format, if any
- Conversation handle
- String handles and their referenced strings
- Transaction-specific data

The following example shows a typical DDESPY display of callback activity:

```
Task:0x8df Time:2882628 Callback:  
Type=Advstart, fmt=0x1("CF_TEXT"), hConv=0xc24b4,  
hsz1=0xc361("System") hsz2=0xc4df("xxcall"), hData=0x0,  
lData1=0x83f0000, lData2=0x0  
return=0x0
```

## Monitoring Errors

### Monitoring Errors

When an error occurs during a DDE transaction, the DDEML places the error value and associated information in a MONERRSTRUCT structure. DDESpy uses this structure to display the following information about the error:

- Task (the handle of the application that caused the error)
- Time
- Error value and name

## Tracking Options

DDESPY can also display information about aspects of DDE communication in your Windows system:

- String handles
- Active conversations
- Active links
- Registered servers

You use the Track menu to specify which DDE activity DDESpy is to track. When you choose a command from the Track menu, DDESpy creates a separate window for the display of information in conjunction with the DDE functions. For each window created, DDESpy updates the displayed information as DDE activity occurs. Events that occurred prior to creation of the tracking window are not displayed in the tracking window.

DDESpy can sort the displayed information in the tracking window. If you select the heading for a particular column in the tracking window, DDESpy will sort the displayed information based on the column you selected. This can be useful if you are searching for a particular event or handle.

### Tracking String Handles

Windows maintains a systemwide string table containing the string handles applications use in DDE transactions. To display the system string table so that the string, the string handle, and the string usage count are shown, choose the String Handles command from the Track menu.

### Tracking Active Conversations

To see a display of all active DDE conversations in your Windows system, choose the Active Conversations command from the Track menu. The Active Conversations window shows the server name, the current topic, and the server and client handles for each active conversation.

### Tracking Active Links

To see a display of all active DDE advise loops, choose the Active Links command from the Track menu. The Active Links window shows the server name, topic, item format, transaction type, client handle, and server handle for every active advise loop in your Windows system.

### Tracking Registered Servers

Server applications use the DdeNameService function to register with the DDEML. When the DDEML receives the DdeNameService function call, it adds the server name and an instance-specific name to a list of registered servers. To see a list of registered servers, choose the Registered Servers command from the Track menu.

## Viewing the Heap: HEAPWALK

The following topics describe how to use HEAPWALK:

[The HEAPWALK Window](#)

[Performing File Operations](#)

[Walking the Heap](#)

[Sorting Memory Objects](#)

[Displaying Memory Objects](#)

[Allocating Memory](#)

[Determining Memory Size](#)

[Suggestions for Using HEAPWALK](#)

## The HEAPWALK Window

### The Heap Walker Window

When you start Heap Walker, it scans the global heap and displays information about the allocated and free memory objects.

Heap Walker displays the following information about each object:

| Column heading | Information displayed                                                                                                                                                                                                                                                                                    |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ADDRESS        | Address of the memory object (displayed in hexadecimal format).                                                                                                                                                                                                                                          |
| HANDLE         | Handle of the memory object (displayed in hexadecimal format).                                                                                                                                                                                                                                           |
| SIZE           | Size of the memory object, in bytes (displayed in decimal format).                                                                                                                                                                                                                                       |
| LOCK           | Lock count of the object. There are two types of lock counts: page-locked (P) and object-locked (L). Page-locked means that virtual memory will not be used to page the object (pieces of the object will not be written to the swap file); object-locked means the entire object will not be discarded. |
| FLG            | D if the object is discardable; F if the object is fixed (not movable or discardable)                                                                                                                                                                                                                    |
| HEAP           | .                                                                                                                                                                                                                                                                                                        |
| OWNER          | Y if the object has a local heap.                                                                                                                                                                                                                                                                        |
| TYPE           | Owner of the object (name of the module that allocated the object).                                                                                                                                                                                                                                      |
|                | Type of object (code segment, data segment, resource, and so on). Heap Walker searches for symbol files and lists names for segments whenever corresponding symbol files are found.                                                                                                                      |



## Performing File Operations: The File Menu

The following commands are on the File menu:

| Command | Action                                                                                                                                                                                                        |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Save    | Saves in a file the current listing of objects in the heap. Heap Walker writes the first listing you save to the file HWG00.TXT and numbers subsequent files consecutively (HWG01.TXT, HWG02.TXT, and so on). |
| Exit    | Closes Heap Walker.                                                                                                                                                                                           |
| About   | Displays information about the current version of Heap Walker.                                                                                                                                                |

When you save a current heap listing to a file, Heap Walker includes all the information shown in the HeapWalker-[Main Heap] window, the number of free blocks in the heap, the size of the largest free block, the total free global heap space, and the following information about each module that has allocated memory from the global heap:

- Module name
- Number of discardable segments loaded in memory
- Number of bytes in discardable segments
- Number of bytes in nondiscardable segments
- Total number of bytes used by the module

## Walking the Heap: The Walk Menu

The following commands are on the Walk menu:

| Command           | Action                                                                                                                                                                                                                               |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Walk Heap         | Displays all objects in the global heap.                                                                                                                                                                                             |
| Walk LRU List     | Displays only discardable objects. Heap Walker lists objects from the least recently used to the most recently used. The object at the top of the list has been least recently used and, therefore, is most eligible for discarding. |
| Walk Free List    | Displays only free blocks of memory.                                                                                                                                                                                                 |
| GC(0) and Walk    | Compacts the global heap, asking for 0 bytes, and then displays the heap.                                                                                                                                                            |
| GC(-1) and Walk   | Attempts to discard all discardable objects and then displays the heap.                                                                                                                                                              |
| GC(-1) and Hit A: | Attempts to discard all discardable objects and then accesses drive A. This command is used to test critical error handling.                                                                                                         |
| Set Swap Area     | Resets the code fence. The code fence defines an area of memory reserved for discardable code.                                                                                                                                       |
| Segmentation Test | Dumps the heap to a file called HWGxx.TXT and then compacts the heap.                                                                                                                                                                |

## Sorting Memory Objects: The Sort Menu

The Sort menu is useful for sorting memory objects in a variety of ways. The following commands are on the Sort menu:

| Command           | Action                                                                                                                                                 |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| Address           | Sorts numerically by address.                                                                                                                          |
| Module            | Sorts alphabetically by the owning module's name and sorts alphabetically by object type within each owner name.                                       |
| Size              | Sorts numerically by object size.                                                                                                                      |
| Type              | Sorts alphabetically by object type and sorts alphabetically by owner name within each object type.                                                    |
| Refresh Seg Names | Searches for symbol files and lists segment names. This command can be used to list segment names for applications loaded after you start Heap Walker. |

## Displaying Memory Objects: The Object Menu

The Object menu is useful for viewing objects selectively. The following commands are on the Object menu:

| Command              | Action                                                                                                                                                                                     |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Show                 | Displays the contents of a selected object in hexadecimal format and ASCII format. When possible for resources, this command displays the resource (such as an icon, menu, or dialog box). |
| Discard              | Discards a selected object.                                                                                                                                                                |
| Oldest               | Marks a selected object as the next candidate for discarding.                                                                                                                              |
| Newest               | Marks a selected object as the last candidate for discarding.                                                                                                                              |
| LocalWalk            | Displays the local heap of the currently selected object, if it has one.                                                                                                                   |
| LC(-1) and LocalWalk | Compacts the selected local heap and then displays the heap.                                                                                                                               |
| GDI LocalWalk        | Displays the GDI local heap and provides information about the objects in the heap.                                                                                                        |
| User LocalWalk       | Displays the USER local heap and provides information about the objects in the heap.                                                                                                       |

### The Show Command

To display a hexadecimal dump of an object, select the object in the HeapWalker-[Main Heap] window and either double-click the left mouse button or choose the Show command from the Object menu. In addition to the hexadecimal dump, the Show command can display the following kinds of resources:

- Bitmaps
- Cursors
- Dialog boxes
- Icons
- Menus

For example, the following illustration shows how the Show command displays the memory and icon associated with the selected memory object.

## The LocalWalk Commands

You can choose the LocalWalk command from the Object menu to view the local heap for a selected object. You can also choose the GDI LocalWalk or User LocalWalk command to view the GDI or USER local heap, respectively, at any time. Local Walk windows show the following information:

|                |                                                                                                                                                                    |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Window heading | Information displayed                                                                                                                                              |
| OFFSET         | Offset of the object from the beginning of the heap. You can use this information to locate the contents of the object within the hexadecimal display of the heap. |
| HANDLE         | Handle of the object.                                                                                                                                              |
| SIZE           | Size of the object, in bytes.                                                                                                                                      |
| FLAGS          | Whether the object is movable, fixed, or free.                                                                                                                     |
| LCK            | Lock count for the object.                                                                                                                                         |
| TYPE           | Object type (shown only for GDI and USER heaps).                                                                                                                   |

The following illustration shows a Local Walk window.

Windows allocates the first object in the local heap, so there are always at least two objects in a local heap.

### Local Walk: The Heap Menu

Following are the commands on the Heap menu:

| Command | Action                                                                                                                                                                                                                                                                                                                  |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Info    | Displays a message box showing the number of free, movable, and fixed objects; the number of bytes they use; the total number of allocated objects; and the number of bytes they use.                                                                                                                                   |
| Save    | Saves the Local Walk display to a file. The first file saved is named HWL00.TXT; subsequent files are numbered sequentially (HWL01.TXT, HWL02.TXT, and so on). The file contains all the information shown in the Local Walk window and a summary of local objects by type (free, movable, fixed, and total allocated). |

### Local Walk: The Sort Menu

Following are the commands on the Sort menu:

| Command | Action                                                                          |
|---------|---------------------------------------------------------------------------------|
| Address | Sorts the Local Walk display numerically by address.                            |
| Flags   | Sorts the Local Walk display alphabetically by flags (fixed, free, or movable). |
| Size    | Sorts the Local Walk display numerically by object size.                        |

### Local Walk: The Add! Menu

The Add! menu displays a message box showing the total number of bytes used by selected objects.

## Allocating Memory: The Alloc Menu

The Alloc menu is useful for allocating memory for test purposes. You can allocate all free memory and then run your program to see how it behaves when no memory is available. You can free all or a specified part of the allocated memory.

The following commands are available from the Alloc menu:

| Command                | Action                                                                                                                                                               |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Allocate All of Memory | Allocates all free memory. This command is useful for testing out-of-memory conditions in applications.                                                              |
| Free All               | Frees memory allocated by the Allocate All of Memory command.                                                                                                        |
| Free 1K                | Frees 1K of the memory allocated by the Allocate All of Memory command.                                                                                              |
| Free 2K                | Frees 2K of the memory allocated by the Allocate All of Memory command.                                                                                              |
| Free 5K                | Frees 5K of the memory allocated by the Allocate All of Memory command.                                                                                              |
| Free 10K               | Frees 10K of the memory allocated by the Allocate All of Memory command.                                                                                             |
| Free 25K               | Frees 25K of the memory allocated by the Allocate All of Memory command.                                                                                             |
| Free 50K               | Frees 50K of the memory allocated by the Allocate All of Memory command.                                                                                             |
| Free XK                | Frees a specified number of kilobytes of the memory allocated by the Allocate All of Memory command. A dialog box is displayed, in which you can specify the number. |

The last eight commands apply only to memory allocated when you chose the first command--it is not possible to free memory allocated by another program.

## Determining Memory Size: The Add! Menu

The Add! menu on the Heap Walker menu bar adds the total number of bytes of selected memory objects. Opening this menu displays a dialog box that shows the number of selected segments and total segment sizes.

## Suggestions for Using HEAPWALK

### Suggestions for Using Heap Walker

One error that frequently occurs in applications is the failure to free memory objects when they are no longer needed. This can cause Windows to fail when one of its data segments grows beyond the 64K limit.

You can use Heap Walker to help determine if your application is not freeing memory objects. With Heap Walker, you can view changes in the sizes of all Windows data segments to observe the effect your application has on these segments.

To check how your application changes the sizes of the Windows data segments, follow these steps:

- 1 Make sure that your application does not generate fatal exits.
- 2 Start the debugging version of Windows.
- 3 Start Heap Walker, and note the sizes of the GDI and USER data segments. This establishes the reference for comparing the size of the data segments later.
- 4 From the Object menu, choose the GDI LocalWalk command to display the GDI Heap (Local Walk) window, which lists the different objects in the GDI data segment. Then choose the Save command from the Heap menu to copy this list to a file; the file will also contain a summary of GDI objects.
- 5 Run your application, and exercise it fully over a long period of time, noting the changes in the size of the GDI and USER data segments that Heap Walker displays as your application runs. While your application is running, repeat step 4 a number of times to take "snapshots" of the effect your application has on the GDI data segment.
- 6 Close your application, take a final snapshot of the GDI data segment, and note the total sizes of the GDI and USER data segments.

As you analyze the data that you've recorded, you should look for GDI objects that your application creates but does not delete when they are no longer needed.



## Monitoring Messages: SPY

Microsoft Windows Spy (SPY.EXE) is a tool for the Microsoft Windows operating system. Spy makes it possible for you to monitor messages sent to one or more windows and to examine the values of message parameters.

Note: If you are using the Microsoft CodeView for Windows (CVW) debugger to debug your application, you can use CVW instead of Spy to trace messages.

The following topics describe how to use SPY:

Choosing Options

Choosing a Window: The Window Menu

Turning Spy On and Off: The Spy Menu

This topic describes how to use the Options!, Window, and Spy menus to specify how Spy is to operate.

## Selecting Options: The Options! Menu

The Options! menu displays a dialog box in which you make selections about the following:

Monitored message types

Output device

Synchronous or asynchronous output

## Selecting Message Types

### Selecting Message Types

Under Messages, you can select any of the following message types you want Spy to monitor:

| Message    | Description                                                                     |
|------------|---------------------------------------------------------------------------------|
| Mouse      | Mouse messages, such as <u>WM_MOUSEMOVE</u> and <u>WM_SETCURSOR</u>             |
| Input      | Input messages, such as <u>WM_CHAR</u> and <u>WM_COMMAND</u>                    |
| System     | Systemwide messages, such as <u>WM_ENDSESSION</u> and <u>WM_TIMECHANGE</u>      |
| Window     | Window manager messages, such as <u>WM_SIZE</u> and <u>WM_SHOWWINDOW</u>        |
| Init       | Initialization messages, such as <u>WM_INITMENU</u> and <u>WM_INITDIALOG</u>    |
| Clipboard  | Clipboard messages, such as <u>WM_RENDERFORMAT</u>                              |
| Other      | Messages other than the types explicitly listed                                 |
| DDE        | Dynamic data exchange (DDE) messages, such as <u>WM_DDE_REQUEST</u>             |
| Non Client | Windows nonclient messages, such as <u>WM_NCDESTROY</u> and <u>WM_NCHITTEST</u> |

By default, Spy monitors all messages.

## Selecting the Output Device

### Selecting the Output Device

Under Output, you can select which of the following output devices you want Spy to send messages to:

| Device | Description                                                                                                                                                                                                                                                                                        |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Window | Spy displays messages in the Spy window. You can specify how many messages Spy stores in its buffer. By default, Spy stores up to 100 lines of messages, which you can view by scrolling through the Spy window. You can also change the maximum number of lines that can be stored in the buffer. |
| Com1   | Spy sends messages to the COM1 port.                                                                                                                                                                                                                                                               |
| File   | Spy sends messages to the specified file. The default output file is SPY.OUT.                                                                                                                                                                                                                      |

## Selecting Frequency of Output

### Selecting Frequency of Output

Under Display, you can select which of the following frequency options you want Spy to use:

| Option       | Description                                |
|--------------|--------------------------------------------|
| Synchronous  | Spy displays messages as it receives them. |
| Asynchronous | Spy queues messages for display.           |

By default, Spy sends messages synchronously.

## Selecting a Window: The Window Menu

Use the Window menu to select the window you want Spy to monitor. The Window menu contains the following commands:

| Command      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Window       | Specifies the window that Spy is to monitor. When you choose the Window command, Spy displays the Spy Window dialog box. This dialog box displays information about the window in which the cursor is located. As you move the cursor from window to window, the following information is updated:<br>Item      Description<br>Window    Handle of the window.<br>Class      Window class.<br>Module    Program that created the window.<br>Parent     Handle of the parent window and the name of the program that created the parent window.<br>Rect       Upper-right and lower-left coordinates of the window and the window size in screen coordinates.<br>Style      Style bits of the window in which the cursor is located, the principal style of the window, and an identifier if the window is a child window. The principal style can be <u>WS_POPUP</u> , <u>WS_ICONIC</u> , <u>WS_OVERLAPPED</u> , or <u>WS_CHILD</u> . |
| All Windows  | Specifies that Spy is to display messages received by all windows.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Clear Window | Clears the Spy window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

## Turning Spy On and Off: The Spy Menu

### Starting and Stopping Spy: The Spy Menu

After using the Options! and Window menus to make your selections, start Spy by clicking the window you selected and choosing the OK button in the dialog box.

To stop monitoring messages, resume monitoring messages, or close Spy, use the Spy menu. The Spy menu contains the following commands:

| Command    | Description                                                  |
|------------|--------------------------------------------------------------|
| Spy On/Off | Starts and stops message monitoring.                         |
| Exit       | Closes Spy.                                                  |
| About Spy  | Provides information about the version of Spy you are using. |

## Compressing Files: compress.exe

### Compressing Files: Compress

Compress (COMPRESS.EXE) creates compressed versions of one or more files. The resulting files are typically 25 to 45 percent smaller than the original files.

Command-line syntax for Compress is as follows:

```
compress [/?][/r] source destination
```

Following are command-line options and parameters for Compress:

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /?          | Displays information about how to use Compress.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| /r          | Specifies that compressed files should be renamed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| source      | Specifies the source filename. The name can include a drive letter, a directory path, or both; and it can contain wildcards.                                                                                                                                                                                                                                                                                                                                                                                                   |
| destination | Specifies the destination. This parameter can consist of a directory (with optional drive letter), a filename, or any combination of the two.<br>If the source parameter contains wildcards and the destination parameter does not specify only a directory, the /r option must be used.<br>If the destination parameter does not contain a filename, Compress uses the filename or filenames specified by the source parameter when Compress copies the file or files to the location specified by the destination parameter. |

The Microsoft File Expansion Utility (Expand) restores files previously compressed by the Compress utility.



## Expanding Compressed Files: expand.exe

### Decompressing Compressed Files: Expand

Expand (EXPAND.EXE) decompresses files previously compressed by Compress. Expand restores these files to their original sizes.

Command-line syntax for Expand is as follows:

```
expand [/?][/r] source destination
```

Following are command-line options and parameters for Expand:

|                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>/?</code>          | Displays information about how to use Expand.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>/r</code>          | Specifies that compressed files should be renamed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>source</code>      | Specifies the source filename. The name can include a drive letter, a directory path, or both; and it can contain wildcards.                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>destination</code> | Specifies the destination. This parameter can consist of a directory (with optional drive letter), a filename, or any combination of the two.<br>If the source parameter contains wildcards and the destination parameter does not specify only a directory, the <code>/r</code> option must be used.<br>If the destination parameter does not contain a filename, Expand uses the filename or filenames specified by the source parameter when Expand copies the file or files to the location specified by the destination parameter. |

The following example shows how to create decompressed versions of all the files on drive A, writing them to a directory on drive C:

```
expand a:*. * c:\mydir
```

## Dr. Watson

Microsoft Windows Dr. Watson is a diagnostic tool for the Microsoft Windows operating system. It detects system and application failures caused by Windows applications and can store information in a disk file. This file can help you find and fix problems in your applications.

Only a single instance of Dr. Watson can be run at a time. Dr. Watson uses the dynamic-link library TOOLHELP.DLL, so it runs only in standard or 386 enhanced mode. Dr. Watson cannot trap faults in a Windows MS-DOS session.

### Configuring Dr. Watson from the WIN.INI File

You can configure Dr. Watson to meet your needs by including settings for any of the following entries in the [Dr. Watson] section of your WIN.INI file (note the space between Dr. and Watson):

DisLen  
DisStack  
GPCContinue  
LogFile  
ShowInfo  
SkipInfo  
TrapZero

## The SkipInfo Entry

### The SkipInfo Entry

The SkipInfo entry controls which parts of the failure report are actually written to disk. Following are the values you can set to disable parts of the failure report:

| Value       | Meaning                                                                                               |
|-------------|-------------------------------------------------------------------------------------------------------|
| 32bitregs   | Disable values of 32-bit registers and of the FS and GS registers on 80386/80486 processors.          |
| clues       | Disable the dialog box titled "Dr. Watson's Clues."                                                   |
| information | Disable system information, such as the Windows version number, processor type, and memory available. |
| registers   | Disable 16-bit registers.                                                                             |
| segments    | Disable segment contents, base addresses, length, and flags.                                          |
| stack       | Disable stack backtrace.                                                                              |
| summary     | Disable four-line summary at beginning of error report.                                               |
| tasks       | Disable list of all active tasks (running applications).                                              |
| time        | Disable Dr. Watson start and stop times.                                                              |

Each of the SkipInfo values can be abbreviated to its first three letters. The following example disables the Dr. Watson's Clues dialog box and the stack backtrace:

```
[Dr. Watson]
```

```
SkipInfo=clu sta
```

## The ShowInfo Entry

### The ShowInfo Entry

Some parts of the Dr. Watson failure report are disabled by default. They can be enabled with the ShowInfo entry. Following are the values you can set to enable parts of the failure report:

| Value       | Meaning                                                                                                                                      |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| disassembly | Enable separate disassembly of the fault address. This does not affect disassembly of stack frames. (See Section 6.1.3, "The DisLen Entry.") |
| errorlog    | Enable error logging.                                                                                                                        |
| locals      | Enable stack dump of local variable and parameter values.                                                                                    |
| modules     | Enable list of all loaded modules, including dynamic-link libraries (DLLs) and font files.                                                   |
| paramlog    | Enable parameter-validation error logging.                                                                                                   |
| sound       | Enable audible warnings.                                                                                                                     |

Each of the ShowInfo values can be abbreviated to its first three letters. The following example sets all six values for the ShowInfo entry, enabling those six parts of the failure report:

```
[Dr. Watson]
```

```
ShowInfo=dis err loc mod par sou
```

## The DisLen Entry

### The DisLen Entry

The DisLen entry controls how many instructions are disassembled in stack traces and the disassembly portion of the failure report. The default value is 8. The following example sets the value to 4:

```
[Dr. Watson]
```

```
DisLen=4
```

## The TrapZero Entry

### The TrapZero Entry

By default, Dr. Watson does not trap divide overflow exceptions, because many applications provide their own handling. The TrapZero entry can be used to enable trapping of divide overflow exceptions, as shown in the following example:

```
[Dr. Watson]
```

```
TrapZero=1
```

## The GPContinue Entry

### The GPContinue Entry

One of the most advanced features of Dr. Watson enables an application to continue even after a general protection (GP) fault occurs. Because a GP fault means that a bug has been encountered, continuing is dangerous. However, some application developers requested the ability to continue running an application even after a GP fault. If the GPContinue entry is used, Dr. Watson performs the following tests when a GP fault occurs. If each of the following four conditions is true, Dr. Watson allows the application to continue:

- 1 Bit 0 of GPContinue is set.
- 2 The faulting instruction is one that can be allowed to continue.  
The following example, which happens to be beyond the end of a segment, would be allowed to continue:  

```
movax, [ffff]
```

The following instruction, which involves an invalid address, would not be allowed to proceed:  

```
jmpseg:offs
```
- 3 The fault is not in KERNEL or USER. (Or the fault is in KERNEL or USER, and you have set the appropriate bit in the GPContinue value to continue in spite of the risk.)
- 4 The user wants to continue. Dr. Watson displays the following dialog box so that the user can decide.

If the user chooses the Close button, an error message box appears.

Although it is very risky, you can also allow continuation in KERNEL or USER by setting GPContinue as required. Following are the bits and values for the GPContinue entry:

| Bit | Value | Meaning                                            |
|-----|-------|----------------------------------------------------|
| 0   | 1     | Allow continuation. (This is the default setting.) |
| 1   | 2     | Write only three-line reports.                     |
| 2   | 4     | Continue even if the fault is in KERNEL.           |
| 3   | 8     | Continue even if the fault is in USER.             |

You must combine these values. The following example permits continuation after a GP fault in USER:

```
set GPContinue=9
```

## The DisStack Entry

### The DisStack Entry

The DisStack entry controls how many levels back on the stack are to be disassembled. The default value is 2. The following example sets the value to 100:

```
[Dr. Watson]
```

```
DisStack=100
```



## The LogFile Entry

### The LogFile Entry

By default, the Dr. Watson log file is named DRWATSON.LOG and placed in the Windows directory. The filename can be changed to any valid filename, even the name of a printer or debugging terminal. For example, to write to a terminal on COM1, use the following setting:

```
[Dr. Watson]
```

```
LogFile=com1
```

## Compiling Resources: Resource Compiler

Microsoft Windows Resource Compiler (RC) is a tool for the Microsoft Windows operating system.

This topic describes how to do the following:

[Including Resources in an Application](#)

[Creating a Resource Script File](#)

[Single-Line Statements](#)

[Multiple-Line Statements](#)

[Directives](#)

[User-Defined Resources](#)

[Using the Resource Compiler](#)

[RC Command-Line Syntax](#)

[Compiling Resources Separately](#)

[Defining Names for the Preprocessor](#)

[Renaming the Compiled Resource File](#)

[Controlling the Directories that RC Searches](#)

[Displaying Progress Messages](#)

## Including Resources in an Application

To include resources in your Windows application, do the following:

- 1 Create individual resource files for cursors, icons, bitmaps, dialog boxes, and fonts. To do this, you can use Microsoft Image Editor and Dialog Editor (IMAGEDIT.EXE and DLGEDIT.EXE) and Microsoft Windows Font Editor (FONTEDIT.EXE).
- 2 Create a resource-definition file that describes all the resources used by the application.
- 3 Use RC to compile the resource-definition file.
- 4 Add the compiled resource files to the application's compiled executable file.

## Creating a Resource-Definition File

After creating individual resource files for your application's icon, cursor, font bitmap, and dialog box resources, you create a resource-definition file. A resource-definition file is an ASCII text file with the file extension `.RC`.

The `.RC` file lists every resource in your application and describes some types of resources in great detail. For a resource that exists in a separate file, such as an icon or cursor, the `.RC` file simply names the resource and the file that contains it. For some resources, such as a menu, the entire definition of the resource exists within the `.RC` file.

An `.RC` file can contain either or both of the following types of information:

- Statements, which name and describe resources.
  - Directives, which instruct RC to perform actions on the resource-definition file before compiling it.
- Directives can also assign values to names.

The following sections describe the statements and directives you can use in a resource-definition file.

## Single-Line Statements

A single-line resource-definition statement can begin with any of the following keywords:

| Keyword              | Description                                                                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b><u>BITMAP</u></b> | Defines a bitmap by naming it and specifying the name of the file that contains it. (To use a particular bitmap, the application requests it by name.) |
| <b><u>CURSOR</u></b> | Defines a cursor by naming it and specifying the name of the file that contains it. (To use a particular cursor, the application requests it by name.) |
| <b>FONT</b>          | Specifies the name of a file that contains a font.                                                                                                     |
| <b>ICON</b>          | Defines an icon by naming it and specifying the name of the file that contains it. (To use a particular icon, the application requests it by name.)    |

## Multiline Statements

A multiline resource-definition statement can begin with any of the following keywords:

| Keyword             | Description                                                                                                               |
|---------------------|---------------------------------------------------------------------------------------------------------------------------|
| <u>ACCELERATORS</u> | Defines menu accelerator keys.                                                                                            |
| <u>DIALOG</u>       | Defines a template that an application can use to create dialog boxes.                                                    |
| <u>MENU</u>         | Defines the appearance and function of an application menu.                                                               |
| <u>RCDATA</u>       | Defines data resources. Data resources let you include binary data directly into the executable file.                     |
| <u>STRINGTABLE</u>  | Defines string resources. String resources are null-terminated ASCII strings that can be loaded from the executable file. |

## Directives

The following directives can be used as needed in the resource-definition file to instruct RC to perform actions or to assign values to names:

| Keyword               | Description                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------------------|
| <code>#define</code>  | Defines a specified name by assigning it a given value.                                         |
| <code>#elif</code>    | Marks an optional clause of a conditional compilation block.                                    |
| <code>#else</code>    | Marks the last optional clause of a conditional compilation block.                              |
| <code>#endif</code>   | Marks the end of a conditional compilation block.                                               |
| <code>#if</code>      | Carries out conditional compilation if a specified expression is true.                          |
| <code>#ifdef</code>   | Carries out conditional compilation if a specified name is defined.                             |
| <code>#ifndef</code>  | Carries out conditional compilation if a specified name is not defined.                         |
| <code>#include</code> | Copies the contents of a file into the resource-definition file before RC processes the latter. |
| <code>#undef</code>   | Removes the current definition of the specified name.                                           |

## Sample Resource-Definition File

The following example shows an .RC file that defines the resources for an application named Shapes:

```
#include "SHAPES.H"

ShapesCursor  CURSOR  SHAPES.CUR
ShapesIcon    ICON    SHAPES.ICO

ShapesMenu    MENU
  BEGIN
    POPUP "&Shape"
    BEGIN
      MENUITEM "&Clear", ID_CLEAR
      MENUITEM "&Rectangle", ID_RECT
      MENUITEM "&Triangle", ID_TRIANGLE
      MENUITEM "&Star", ID_STAR
      MENUITEM "&Ellipse", ID_ELLIPSE
    END
  END
```

The CURSOR statement names the application's cursor resource ShapesCursor and specifies the cursor file SHAPES.CUR, which contains the image for that cursor.

The ICON statement names the application's icon resource ShapesIcon and specifies the icon file SHAPES.ICO, which contains the image for that icon.

The MENU statement defines an application menu named ShapesMenu, a pop-up menu with five menu items.

The menu definition, enclosed by the BEGIN and END keywords, specifies each menu item and the menu identifier that is returned when the user selects that item. For example, the first item on the menu, Clear, returns the menu identifier ID\_CLEAR when the user selects it. The menu identifiers are defined in the application header file, SHAPES.H.

For more information about resource-definition files, the syntax of resource statements, and how to define your own resources, see the Microsoft Windows Programmer's Reference, Volume 4.



Resource Compiler (RC) serves the following functions:

- It compiles the resource-definition file and the resource files (such as icon and cursor files) into a binary resource (.RES) file.
- It combines the .RES file with the executable (.EXE) file created by the linker; the result is an executable Windows application.
- It marks the Windows application as a Windows 3.0 or Windows 3.1 application.

Note: Each Windows application and dynamic-link library (DLL) must be identified with a Windows version number. For this reason, use RC on each Windows application or DLL you build, even if it uses no resources. For more information about Windows versions, see the discussions of the /30 and /31 options in Specifying Options.

## Command-Line Syntax

To start RC, use the rc command. What you need to specify on the command line depends on whether you are compiling resources, adding compiled resources to an executable file, or both.

The following line shows rc command-line syntax:

```
rc [options] definition-file [executable-file]
```

Following are several ways you can use the rc command:

- To compile resources separately, use the rc command in the following form:  
rc /r [options] definition-file  
When you use this form, RC ignores any executable file you specify.
- To compile an .RC file and add the resulting .RES file to the executable file, use the rc command in the following form:  
rc [options] definition-file [executable-file]
- To compile an application or DLL that does not have a .RES file, use the rc command in the following form:  
rc [options] dll-or-executable-file  
When you use this form, the filename must explicitly have an .EXE, .DRV, or .DLL extension.
- To simply add a compiled resource (.RES) file to an executable file, use the rc command in the following form:  
rc [options] res-file.res [executable-file]

## Specifying Options

The rc command's options parameter can include one or more of the following options:

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /30         | Marks the executable file so it will run with Windows version 3.0 or Windows version 3.1. By default, RC marks the executable file to run only with Windows 3.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| /31         | Marks the executable file so it will run only with Windows 3.1. This is the default condition.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| /?          | Displays a list of rc command-line options.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| /d          | Defines a symbol for the preprocessor that you can test with the #ifdef directive.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| /e          | Changes the default location of global memory for a DLL from below the Expanded Memory Specification (EMS) bank line to above the EMS bank line. This option has no effect with Windows 3.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| /fe newname | Uses newname for the name of the .EXE file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| /fo newname | Uses newname for the name of the .RES file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| /h          | Displays a list of rc command-line options.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| /i          | Searches the specified directory before searching the directories specified by the INCLUDE environment variable.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| /k          | Disables the load-optimization feature of RC. If this option is not specified, the compiler arranges segments and resources in the executable file so that all preloaded information is contiguous.<br><br>This feature allows Windows to load the application much more quickly. If you do not specify the /k option, all data segments, nondiscardable code segments, and the entry-point code segment will be preloaded, unless any segment and its relocation information exceed 64K. If the PRELOAD attribute is not assigned to these segments in the module-definition (.DEF) file when you link your application, RC will add the PRELOAD attribute and display a warning. Resources and segments will have the same segment alignment. This alignment should be as small as possible to limit the size of the final executable file. You can set the alignment by using the link command with the /a option. |
| /l[im32]    | Specifies to Windows that the application uses expanded memory directly, according to the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS), version 3.2. This option has no effect with Windows 3.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| /m[ultinst] | Assigns each instance of the application task to a distinct EMS bank when Windows is running with the EMS 4.0 memory configuration. (By default, all instances of a task share the same EMS bank.) This option has no effect with Windows 3.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| /p          | Creates a private DLL that is called by only one application. This allows Windows to use memory more efficiently, because only one application (or multiple instances of the same application) calls the DLL. For example, in the large-frame EMS memory model,                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

the DLL is loaded above the EMS bank line, freeing memory below the bank line. This option has no effect with Windows 3.1.

- `/r` Creates an .RES file from an .RC file. Use this option when you do not want to add the compiled .RES file to the .EXE file.
- `/t` Creates an application that runs with Windows only in protected (standard or 386 enhanced) mode. If the user attempts to run the application in real mode, Windows will display a message that the application cannot run in real mode. This option has no effect with Windows 3.1.
- `/v` Displays messages that report on the progress of the compiler.
- `/x` Prevents RC from checking the INCLUDE environment variable when searching for header files or resource files.
- `/z` Prevents RC from checking for RCINCLUDE statements. When you have not used RCINCLUDE statements, using this option can greatly improve the speed of RC.

Options are not case-sensitive, and a hyphen (-) can be used in place of a slash mark (/). You can combine single-letter options if they do not require any additional parameters. For example, the following two commands are equivalent:

```
RC /R /V SAMPLE.RC
```

```
rc -rv sample.rc
```

#### Specifying the Resource-Definition File

The `rc` command's `definition-file` parameter specifies the name of the resource-definition file that contains the names, types, filenames, and descriptions of the resources to be added to the .EXE file. It can also specify the name of a compiled .RES file, in which case RC adds the compiled resources to the executable file.

#### Specifying the Executable File

The `rc` command's `executable-file` parameter specifies the name of the executable file that the compiled resources should be added to. If you do not specify an executable file, RC uses the executable file with the same name as the resource-definition file (excluding the filename extension).

#### Renaming the Executable File

The `rc` command's `/fe` option makes it possible for you to specify the name of the final executable file. The following example combines MYEXE.EXE with MYRES.RES to produce the final executable file FINAL.EXE:

```
rc /fe final.exe myres.res myexe.exe
```

### Compiling Resources Separately

By default, RC adds the compiled resources to the specified executable file. Sometimes you might want to first compile the resources and then add them to the executable file in separate steps. This can be useful because resource files typically change little after initial development. You can save time by compiling your application's resources separately and then adding the compiled .RES file to your executable file each time you recompile the .EXE file.

You can use the /r option to compile the resources separately without adding them to the executable file. When you use this option, RC compiles the .RC file and creates a compiled resource (.RES) file.

For example, the following command reads the resource-definition file SAMPLE.RC and creates the compiled resource file SAMPLE.RES:

```
rc -r sample.rc
```

In this case, RC does not add SAMPLE.RES to the executable file.

## Defining Names for the Preprocessor

You can specify conditional branching in a resource-definition file, based on whether a term is defined on the rc command line with the /d option.

For example, suppose your application has a pop-up menu, the Debug menu, that should appear only during debugging. When you compile the application for normal use, the Debug menu is not included. The following example shows the statements that can be added to the resource-definition file to define the Debug menu:

```
MainMenu MENU
BEGIN
    .
    .
    .
#ifdef DEBUG
    POPUP "&Debug"
    BEGIN
        MENUITEM "&Memory usage", ID_MEMORY
        MENUITEM "&Walk data heap", ID_WALK_HEAP
    END
#endif
END
```

When compiling resources for a debugging version of the application, you could include the Debug menu by using the following rc command:

```
rc -r -d debug myapp.rc
```

To compile resources for a normal version of the application--one that does not include the Debug menu--you could use the following rc command:

```
rc -r myapp.rc
```

### Renaming the Compiled Resource File

By default, when compiling resources, RC names the compiled resource (.RES) file with the same name as the .RC file (but not the same extension) and places it in the same directory as the .RC file. The following example compiles MYAPP.RC and creates a compiled resource file named MYAPP.RES in the same directory as MYAPP.RC:

```
rc -r myapp.rc
```

The /fo option lets you give the resulting .RES file a name that differs from the name of the corresponding .RC file. For example, to name the resulting .RES file NEWFILE.RES, you would type the following command:

```
rc -r -fo newfile.res myapp.rc
```

The /fo option can also place the .RES file in a different directory. For example, the following command places the compiled resource file MYAPP.RES in the directory C:\SOURCE\RESOURCE:

```
rc -r -fo c:\source\resource\myapp.res myapp.rc
```

## Controlling Which Directories the Resource Compiler Searches

By default, RC searches for header files and resource files (such as icon and cursor files) first in the current directory and then in the directories specified by the INCLUDE environment variable. (The PATH environment variable has no effect on which directories RC searches.)

### Adding a Directory to Search

You can use the /i option to add a directory to the list of directories RC searches. The compiler then searches the directories in the following order:

- 1 The current directory
- 2 The directory or directories you specify by using the /i option, in the order in which they appear on the rc command line
- 3 The list of directories specified by the INCLUDE environment variable, in the order in which the variable lists them, unless you specify the /x option

The following example compiles the resource-definition file MYAPP.RC and adds the compiled resources to MYAPP.EXE:

```
rc /i c:\source\stuff /i d:\resources myapp.rc
```

When compiling the resource-definition file MYAPP.RC, RC searches for header files and resource files first in the current directory, then in C:\SOURCE\STUFF and D:\RESOURCES, and then in the directories specified by the INCLUDE environment variable.

### Suppressing the INCLUDE Environment Variable

You can prevent RC from using the INCLUDE environment variable when determining the directories to search. To do so, use the /x option. The compiler then searches for files only in the current directory and in any directories you specify by using the /i option.

The following example compiles the resource-definition file MYAPP.RC and adds the compiled resources to MYAPP.EXE:

```
rc /x /i c:\source\stuff myapp.rc
```

When compiling the resource-definition file MYAPP.RC, RC searches for header files and resource files first in the current directory and then in C:\SOURCE\STUFF. It does not search the directories specified by the INCLUDE environment variable.

### Displaying Progress Messages

By default, RC does not display messages that report on its progress as it compiles. You can, however, specify that RC is to display these messages. To do so, use the `/v` option.

The following example causes RC to report on its progress as it compiles the resource-definition file `SAMPLE.RC`, creates the compiled resource file `SAMPLE.RES`, and adds the `.RES` file to the executable file `SAMPLE.EXE`:

```
rc /v sample.rc
```



## Creating Help Files

Microsoft Windows Help provides online help for users working with a Windows application. Windows Help provides a practical way to present information about your application in a format users can access easily.

This topic introduces the tools you can use to develop Windows Help files and to incorporate Help in Windows applications.

### About Windows Help Files

Windows Help files can display information by using the following elements:

- Text in multiple fonts, sizes, and colors
- Bitmaps and metafiles with up to 16 colors
- Segmented-graphics bitmaps with embedded hot spots
- Cross-reference jumps for links to additional information
- Pop-up windows to present text and graphics
- Secondary windows to present information without the full menus and buttons of Windows Help
- Keywords to help users find the information they need

You create help files by creating topic and graphics files and a Help project file. A topic file contains the text for the help topic and contains the Help statements and macros that define the format of the text and the position of graphics in each topic. The graphics files contain the bitmaps and metafiles you want to display in the topics. The project file contains a description of how to build the help file.

You use the Microsoft Help Compiler to build the final help file. Combining the topic, graphics, and project files, the compiler creates a single help file (with the filename extension .HLP) that you can open and view by using Windows Help.

For more information about creating help files, see the following topics:

[Creating Topic Files](#)

[Using Graphics Files](#)

[Creating Help Project Files](#)

[Using Help in a Windows Application](#)

[Help Macros](#)

[HPJ Statements](#)

[RTF Tokens](#)

## Creating Topic Files

A topic file contains the text for the help file, as well as the statements and macros that define the format of the text and the position of the graphics. Every topic file consists of one or more topics. A topic is any distinct unit of information, such as a contents screen, a conceptual description, a set of instructions, a keyboard table, a glossary definition, a list of jumps, a picture, and so on.

Windows Help displays only one topic at a time, but a user can view any topic in a help file by using a link to the topic or searching for keywords associated with the topic.

You create topic files directly by using a text editor and inserting Help statements. You can create them indirectly by using a word processor that generates rich-text format (RTF) files. The Help statements are an extended subset of the RTF statements, which provide a wide variety of formatting capabilities.

### Declaring Character Set, Fonts, and Colors

When you create a topic file, you must ensure that the entire contents of the file are enclosed in braces ( { } ). The first statement in the file must be the `\rtf` statement; it immediately follows the first opening brace. You should follow the `\rtf` statement with a `\ansi` statement (or a similar statement) that specifies the character set used in the file. The following example shows the general form for a topic file:

```
{\rtf1\ansi
.
.
.
}
```

You must declare the names of the fonts you use in the file by using a `\fonttbl` statement. The `\fonttbl` statement, enclosed in braces, contains a list of font and family names and specifies a unique number for each font. You use these numbers with `\f` statements later in the file to set specific fonts. The following `\fonttbl` statement assigns font numbers 0, 1, and 2 to the TrueType fonts Times New Roman®, Courier New®, and Arial®, respectively:

```
{\fonttbl
\f0\froman Times New Roman;
\f1\fdecor Courier New;
\f2\fswiss Arial;}
}
```

You should also use the `\deff` statement to set the default font for the file. Windows Help uses this default font if no other font is specified. The following example sets the default font number to zero, corresponding to the Times New Roman font specified in the previous `\fonttbl` statement:

```
\deff0
```

If you use specific text colors or choose not to rely on the default text colors set by Windows, you must define your colors by using a `\colortbl` statement. The `\colortbl` statement, enclosed in braces, defines each color by specifying the amount of each primary color (red, green, and blue) used in it. The statement implicitly numbers the colors consecutively starting from zero. You use these color numbers with `\cf` statements later in the file to set the color. The following example creates four colors (black, red, green, and blue):

```
{\colortbl
\red0\green0\blue0;
\red255\green0\blue0;
\red0\green128\blue0;
\red0\green0\blue255;}
}
```

Although it is not shown here, you can put a semicolon immediately after the `\colortbl` statement to define the default color as 0.

### Defining Individual Topics

Each topic starts with one or more `\footnote` statements and ends with a `\page` statement. All text and graphics specified between these statements belong to the topic.

Every topic must have a context string. Windows Help uses the context string to locate the topic when the user requests to view it. You assign a context string to a topic by using the `\footnote` statement and the number sign (#) footnote character. Context strings can consist of letters, digits, and the underscore character (\_). To prevent conflicts, each context string in a help file must be unique.

You can also assign a title to the topic by using the `\footnote` statement and the dollar sign (\$) footnote character. Windows Help uses the title to identify the topic in the History and Search dialog boxes. You must provide a title if you assign keywords to the topic.

The following example defines a small topic having the context string "topic1" and the title My Topic:

```
#{\footnote topic1}
$(DOLLAR_BRACE)\footnote My Topic}
This is my first topic.
\par
\page
```

In general, you use the `\par` statement to mark the end of each paragraph. In this example, the `\par` statement marks the end of the only paragraph in the topic.

You can add a macro to a topic by using the `\footnote` statement and the exclamation point (!) as the footnote character. For example, the following `\footnote` statement adds the CopyTopic macro to the topic:

```
!{\footnote CopyTopic() }
```

Windows Help executes the macro each time it displays the topic.

The total size of text and graphics data stored in a paragraph must not exceed 64K. (Bitmaps included by using the `bmc`, `bml`, and `bmr` statements do not contribute to this total.)

#### Setting Font Size and Name

You can set the font name and size by using the `\f` and `\fs` statements. The name is set by using a font number specified in the `\fonttbl` statement. The size of the font is specified in half-points. The following example sets the text to 10-point Times New Roman (if the `\fonttbl` statement matches the example given earlier):

```
\f0\fs20
```

Once you set the font name and size, the settings apply to all subsequent text up to the next `\plain` statement or until you change the name or size by using the `\f` or `\fs` statement again. The `\plain` statement resets the name and font to the defaults. The default font name is as set by the `\deff` statement; the default font size is 12 points.

#### Setting Space Before and After Paragraphs

You can set the amount of space before and after each paragraph by using the `\sb` and `\sa` statements. These statements let you control the amount of space that appears between paragraphs. You specify the space in twips. (A twip is 1/1440 inch, or 1/20 of a printer's point). The following example sets the space before a paragraph to 360 twips:

```
\sa360
This paragraph has 360 twips space immediately before it.
\par
This paragraph also has 360 twips before it.
\par
```

Once you set the space before or after a paragraph, the spacing applies to all subsequent paragraphs up to the next `\pard` statement or until you change the spacing by using the `\sa` and `\sb` statements again. The `\pard` statement restores the default spacing.

#### Setting the Left and Right Indents

When Windows Help displays its window, it automatically creates left and right margins and wraps text to fit within these margins. The margins are positioned slightly within the left and right edges of the window to prevent text in the topic from being clipped by the window.

You can override these margins by setting the left and right indents for a paragraph. The `\li` and `\ri` statements set an indent to a position relative to the corresponding left and right margins. For example, the following paragraph is indented 1 inch (1440 twips) from the left margin:

```
\\li1440
This paragraph is indented 1 inch.
\par \pard
This paragraph is not indented.
```

Once indents are set, they apply to all subsequent paragraphs up to the next `\pard` statement. Note that the `\pard` statement must follow the `\par` statement that ends the paragraph to be indented.

You can set an indent for the first line in a paragraph by using the `\fi` statement. This allows you to create paragraphs with hanging indents. It is also useful for creating two-column lists.

#### Setting Tab Stops

You can set tab stops by using the `\tx` statement. You can use one or more `\tx` statements, each setting a specific position in twips relative to the left margin. Once you have set tab stops, you can use the `\tab` statement to align subsequent text with the next tab. The tab settings remain active until you use the `\pard` statement. The following example creates a two-column list by using a tab stop and paragraph indenting:

```
\fi-1440\li1440\tx1440
left
\tab
right
\par
left
\tab
right
\par
\pard
```

#### Breaking Lines

Ordinarily, Windows Help wraps all lines in a paragraph, fitting as many words on a line as will fit between the current left and right indents. You can force Windows Help to break a line at a given place by using the `\line` statement. You can control wrapping by using the `\keep` and `\pard` statements.

The following example uses the `\keep` statement to turn off word wrapping for three short lines and uses the `\pard` statement to restore the default properties:

```
\keep
3 pairs black socks\line
5 pairs blue socks\line
2 pairs brown socks\line
\par
\pard
```

The following example uses the `\keep` and `\pard` statements to create three nonwrapping paragraphs:

```
\keep
3 pairs black socks
\par
5 pairs blue socks
\par
2 pairs brown socks
\par
\pard
```

#### Creating Links and Pop-up Topics

Windows Help displays only one topic at a time. To enable users to view other topics, you must create hot spots that link your topics to other topics. You create a hot spot by using the `\strike`, `\ul`, or `\uldb` statement and a corresponding `\v` statement. When you create a link, you provide the text for the hot spot and the context string for the topic that is to be jumped to or displayed. The following example creates a hot spot named Glossary and establishes a link from the hot spot to the topic having the context string "glo1":

You can find a list of terms used in this help file in the `{\uldb Glossary}{\v glo1}`.

When Windows Help displays the topic with this hot spot, it places a line under the word Glossary and colors the word green. The context string is not shown, but if the user clicks on the hot spot, Windows Help jumps to and displays the corresponding topic.

The `\strike` and `\uldb` statements are used to create jumps to other topics. The `\ul` statement creates a link to

a pop-up topic. Windows Help displays pop-up topics in a pop-up window and leaves the current topic in the main window.

You can also associate a Help macro with a hot spot in a topic. For example, the following `\uldb` and `\v` statements create a hot spot for the `ExecProgram` macro:

```
{\uldb Clock}{\v !ExecProgram("clock.exe", 1)}
```

Windows Help executes the macro whenever the user chooses the hot spot. Windows Help continues displaying the topic while it executes the macro, unless the macro causes a jump to another topic.

#### Creating a Keyword List

You can also enable users to find and view topics by assigning keywords to the topics. You assign a keyword by using the `\footnote` statement and the letter `K` as the footnote character. Windows Help collects all keywords in a help file and displays them in its Search dialog box. Using this dialog box, a user can select a keyword and view the help topics associated with it. The following example assigns the keyword "Sample Topics" to the current topic:

```
#{\footnote topic1}  
$(DOLLAR_BRACE)\footnote My Topic}  
K{\footnote Sample Topics}  
This is my first topic.  
\par  
\page
```

If a keyword begins with the letter `K`, you must place an extra space before the word. Multiple keywords for a topic are separated by semicolons.

A keyword can be assigned to any number of topics. When the user selects the keyword in the Search dialog box, Windows Help displays all topics associated with the keyword. The user then picks the one to view.

You can also create alternative keywords for a help file for use with the `WinHelp` function.

#### Creating Browse Sequences

You can enable users to browse through a sequence of help topics by creating a browse sequence and adding browse buttons to your help file. A browse sequence typically consists of two or more related topics that are intended to be read sequentially. You create a browse sequence by using the `\footnote` statement and the plus-sign (+) footnote character to assign a sequence identifier. The following example assigns a sequence identifier to the topic titled A Topic:

```
#{\footnote topic5}  
$(DOLLAR_BRACE)\footnote A Topic}  
+{\footnote shorttopics}  
This is one topic in a browse sequence.  
\par  
\page
```

Windows Help adds topics with sequence identifiers to the browse sequence and determines the order of topics in the sequence by sorting the identifiers alphabetically. If two topics have the same identifier, Windows Help assumes that the topic that was compiled first is to be displayed first.

Windows Help uses the sequence only if the browse buttons have been enabled. You can enable the buttons by placing the following statements in the Help project file:

```
[CONFIG]  
BrowseButtons( )
```

For more information about the project file, see Section 3.4, "Creating Help Project Files."

You can create more than one browse sequence in a help file by using sequence numbers with sequence identifiers. The sequence number consists of a colon (:) followed by an integer. Windows Help combines all topics having the same sequence identifier (but different sequence numbers) into a single browse sequence and determines the order of the topics by sorting them alphabetically. To ensure that numerals are sorted correctly, they should have the same number of digits. For example, the numerals 1 through 10 should be 01 through 10.

```
#{\footnote topic10}  
$(DOLLAR_BRACE)\footnote Alpha Topic #3}
```

```
+{\footnote alpha:3}
This topic is part of the alpha browse sequence.
\par
\page
```

## Using Graphics Files

You can add bitmaps and metafiles to your help files by using the `bml`, `bmc`, and `bmr` statements. These statements take the name of a graphics file and insert the corresponding bitmap or metafile into the help file at the specified position.

Windows Help requires graphics files to be in one of the following formats:

- Windows bitmap (.BMP)
- Placeable Windows metafile (.WMF)
- Multiple-resolution bitmap (.MRB)
- Segmented-graphics bitmap (.SHG)

Multiple-resolution bitmaps can be created by using the Microsoft Multiple-Resolution Bitmap Compiler (MRBC). Segmented-graphics bitmaps can be created by using Microsoft Windows Hotspot Editor. Only 16-color and monochrome bitmaps may be used. Windows Help does not support 256-color bitmaps.

Although the `\pict` statement can also be used to add bitmaps and metafiles to a help file, the bitmap or metafile data must be inserted into the topic file rather than specified as a separate file.

### Inserting a Bitmap in Text

You can insert a bitmap into a paragraph as if it were a character by using the `bmc` statement. The statement aligns the bottom of the bitmap with the base line of the current line of text and places the left edge of the bitmap at the next character position.

Since the bitmap is treated as text, any paragraph properties assigned to the paragraph also apply to the bitmap. Windows Help places text following the bitmap on the same base line at the next available character position.

In general, bitmaps inserted as characters should be clipped to the smallest possible size. Any extra white space at the top or bottom of the bitmap image affects the alignment of the bitmap with the text and may affect the spacing between lines.

You must not specify negative line spacing for paragraphs that contain `bmc` statements. Doing so might cause the bitmap to appear on top of the paragraph.

### Wrapping Text Around a Bitmap

You can place a bitmap at the left or right margin of the Help window and have subsequent text wrap around the bitmap by using the `bml` or `bmr` statement. The `bml` statement inserts a bitmap at the left margin; `bmr` inserts it at the right.

If you want text to wrap around a bitmap, you must place the `bml` or `bmr` statement at the beginning of a paragraph. Windows Help aligns the start of the paragraph with the top of the bitmap and wraps around the left or right edge of the bitmap.

If you place a `bml` or `bmr` statement at the end of a paragraph, Windows Help places the bitmap under the paragraph instead of wrapping the text around the bitmap. If you do not want text to wrap around a bitmap, place `\par` statements immediately before and after the `bml` or `bmr` statement.

### Using a Bitmap as a Hot Spot

You can use bitmaps as hot spots. This enables you to create graphics, such as icons or buttons, and use them as "jumps" to particular topics or as hot spots for macros.

You can also divide a single bitmap into several hot spots and assign a different link or macro to each hot spot. Such bitmaps, called segmented-graphics bitmaps, are created by using Hotspot Editor. For example, if you have a bitmap of a dialog box, you can assign links to each of the control windows in the dialog box, enabling the user to click a control window and view information about it. Segmented-graphics bitmaps already contain the context strings needed for the links; only a `bml` or `bmr` statement is needed to insert the bitmap. The `\strike` and `\v` statements must not be used.

### Using a Bitmap on Different Displays

A multiple-resolution bitmap is a single bitmap file that contains one or more bitmaps that have been marked for use with specific displays, such as the CGA, EGA, VGA, or 8514 displays. You use multiple-resolution bitmaps to avoid problems associated with displaying bitmaps designed for a single type of display. Single-resolution bitmaps can have the following problems:

- Appear too big or too small on displays having different resolutions
- Appear stretched or compressed on displays with different aspect ratios
- Lack colors or use unintended colors on displays with different color capabilities.

You create multiple-resolution bitmaps by using MRBC. The compiler, an MS-DOS program, has the following command-line syntax:

```
mrbc [/s] filename ...
```

The filename parameter specifies the name of a Windows bitmap file. Typically, you specify several filenames, one for each type of display. Wildcards can be used. The compiler uses the filename of the first bitmap file as the name of the output file but gives the output file the filename extension .MRB. The following example combines the bitmap files MYBUTTON.EGA, MYBUTTON.VGA, and MYBUTTON.854 into the multiple-resolution bitmap file MYBUTTON.MRB:

```
mrbc mybutton.ega mybutton.vga mybutton.854
```

In this example, the compiler checks the `biXPelsPerMeter` and `biYPelsPerMeter` members of the `BITMAPINFOHEADER` structure in each bitmap file to determine the display type for the bitmap. (For a description of the `BITMAPINFOHEADER` structure, see the Microsoft Windows Programmer's Reference, Volume 3.) If these members are set to zero, the compiler prompts for the display type with a message such as the following:

```
Please enter the monitor type for the bitmap mybutton.ega:
```

You must enter at least the first character of one of the following display-type names: CGA, EGA, VGA, or 8514. The compiler sets the display type you specify, but it does not check that the type is valid. For example, if you specify VGA for an EGA bitmap, the compiler marks it as a VGA bitmap. The result may be undesirable.

The `/s` option, specifying silent mode, speeds up compilation if the names of the bitmap files conform to the MRBC filename convention. If you use the `/s` option, the compiler uses the first character of the filename extension to determine the display type for the bitmap, as described in the following list:

| Letter | Meaning     |
|--------|-------------|
| C      | CGA bitmap  |
| E      | EGA bitmap  |
| V      | VGA bitmap  |
| 8      | 8514 bitmap |

If the filename extension starts with any other character, MRBC assumes a VGA bitmap. The following example creates the multiple-resolution bitmap file MYBUTTON.MRB, containing bitmaps for EGA, VGA, and 8514 displays:

```
mrbc /s mybutton.ega mybutton.vga mybutton.854
```

The compiler never writes over existing multiple-resolution bitmap files. If the output file already exists, the compiler displays an error message.

You insert multiple-resolution bitmaps into your help file by using the same statements as for Windows bitmaps.

Before displaying a multiple-resolution bitmap, Windows Help checks the display type for the computer and then selects the bitmap that has the closest matching resolution, aspect ratio, and color capabilities. Windows Help never displays more than one bitmap from a multiple-resolution bitmap file.



## Creating Help Project Files

This section describes the format and contents of the Help project file (.HPJ) used to build the help file. The project file contains all the information the Microsoft Help Compiler needs to combine topic files and other elements into a help file.

### Project File Sections

Every project file consists of one or more sections. Each section has a section name, enclosed in brackets ([ ]), that defines the purpose and format of statements and options in the section. Following are the sections used in project files:

| Section            | Description                                                                                                                                                                                                                     |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>[OPTIONS]</u>   | Specifies options that control the build process. This section is optional. If this section is used, it should be the first section listed in the project file, so that the options will apply during the entire build process. |
| <u>[FILES]</u>     | Specifies topic files to be included in the build. This section is required.                                                                                                                                                    |
| <u>[BUILDTAGS]</u> | Specifies valid build tags. This section is optional.                                                                                                                                                                           |
| <u>[CONFIG]</u>    | Specifies Help macros that define nonstandard menus, buttons, and macros used in the help file. This section is required if the help file uses any of these features. This section is new for Windows 3.1.                      |
| <u>[BITMAPS]</u>   | Specifies bitmap files to be included in the build. This section is not required if the project file lists a path for bitmap files by using the BMROOT or ROOT option.                                                          |
| <u>[MAP]</u>       | Associates context strings with context numbers. This section is optional.                                                                                                                                                      |
| <u>[ALIAS]</u>     | Assigns one or more context strings to the same topic. This section is optional.                                                                                                                                                |
| <u>[WINDOWS]</u>   | Defines the characteristics of the primary Help window and the secondary-window types used in the help file. This section is required if the help file uses secondary windows. This section is new for Windows 3.1.             |
| <u>[BAGGAGE]</u>   | Lists files that are to be placed within the help file (which contains its own file system). This section is optional.                                                                                                          |

Every project file requires a [FILES] section. This section names the topic files. Most project files also have an [OPTIONS] section that specifies how to build the help file. A very useful option in the [OPTIONS] section is the COMPRESS option, which specifies whether the help file should be compressed or uncompressed. Compressing a help file reduces its size considerably and saves valuable disk space.

The following example creates a compressed help file from two topic files, MAIN.RTF and MENUS.RTF:

```
[ OPTIONS ]  
COMPRESS=TRUE  
  
[ FILES ]  
MAIN . RTF  
MENUS . RTF
```

### Using Macros in Project Files

You can add macros to the [CONFIG] section of a project file. Since Windows Help executes the macros when it first opens the help file, macros that create menus, menu items, and buttons are typically placed in this section. If there is more than one macro listed in the [CONFIG] section, Windows Help executes them in the order in which they are listed.

You can create new menu items and buttons for Windows Help by using such macros as CreateButton and InsertMenu. These macros define other Help macros and associate them with the menu items and buttons. Windows Help executes these macros when the user chooses a corresponding menu item or button. Macros that create Help buttons, menus, or menu items remain in effect until the user quits Windows Help or opens a new help file.

You can extend the capabilities of Windows Help by developing your own dynamic-link libraries (DLLs) and defining Help macros that call functions in the libraries. To define Help macros that call DLL functions, you must register each function and its corresponding library by using the RegisterRoutine macro in the [CONFIG] section of the project file.

### Sample Project File

The following example is a sample project file for the Cardfile application. Comments, marked by a beginning semicolon (;), indicate the purpose of each section in the file:

```
; Options used to define the Help title bar and icon
[OPTIONS]
ROOT=C:\HELP
BMROOT=C:\HELP\ART
CONTENTS=cont_idx_card
TITLE=Cardfile Help
ICON=CARDHLP.ICO
COMPRESS=OFF
WARNING=3
REPORT=ON
ERRORLOG=CARD.BUG

; Files used to build Cardfile Help
[FILES]
RTFTXT\COMMANDS.RTF
RTFTXT\HOWTO.RTF
RTFTXT\KEYS.RTF
RTFTXT\GLOSSARY.RTF

; Button macros and Using Help file
[CONFIG]
CreateButton("btn_up", "&Up", "JumpContents(`HOME.HLP')")
BrowseButtons()
SetHelpOnFile("APPHELP.HLP")

; Secondary-window characteristics
[WINDOWS]
picture = "Samples", (123,123,256,256), 0, (0,255,255), (255,0,0)
```

## Using Help in a Windows Application

Windows applications can offer help to their users by using the WinHelp function to start Windows Help and display topics in the application's help file. The WinHelp function gives a Windows application complete access to the help file, as well as to the menus and commands of Windows Help. Many applications use WinHelp to implement context-sensitive Help. Context-sensitive Help enables users to view topics about specific windows, menus, menu items, and control windows by selecting the item with the keyboard or the mouse. For example, a user can learn about the Open command on the File menu by selecting the command (using the direction keys) and pressing the F1 key.

### Choosing Help from the Help Menu

Every application should provide a Help menu to allow the user to open the help file with either the keyboard or the mouse. The Help menu should contain at least one Contents menu item that, when chosen, displays the contents or the main topic in the help file. To support the Help menu, the application's main window procedure should check for the Contents menu item and call the WinHelp function, as in the following example:

```
case WM_COMMAND:
    switch (wParam) {
        case IDM_HELP_CONTENTS:
            WinHelp(hwnd, "myhelp.hlp", HELP_CONTENTS, 0L);
            return 0L;
        .
        .
        .
    }
    break;
```

You can add other menu items to the Help menu for topics containing general information about the application. For example, if your help file contains a topic that describes how to use the keyboard, you can place a Keyboard menu item on the Help menu. To support additional menu items, your application must specify either the context string or the context identifier for the corresponding topic when it calls the WinHelp function. The following example uses a Help macro to specify the context string IDM\_HELP\_KEYBOARD for the Keyboard topic:

```
case IDM_HELP_KEYBOARD:
    WinHelp(hwnd, "myhelp.hlp", HELP_COMMAND,
        (LPSTR)"JumpID(\"myhelp.hlp\", \"IDM_HELP_KEYBOARD\")");
    return 0L;
```

A better way to display a topic is to use a context identifier. To do this, the help file must assign a unique number to the corresponding context string, in the [MAP] section of the project file. For example, the following section assigns the number 101 to the context string IDM\_HELP\_KEYBOARD:

```
[MAP]
IDM_HELP_KEYBOARD    101
```

An application can display the Keyboard topic by specifying the context identifier in the call to the WinHelp function, as in the following example:

```
#define IDM_HELP_KEYBOARD 101

WinHelp(hwnd, "myhelp.hlp", HELP_CONTEXT, (DWORD)IDM_HELP_KEYBOARD);
```

To make maintenance of an application easier, most programmers place their defined constants (such as IDM\_HELP\_KEYBOARD in the previous example) in a single header file. As long as the names of the defined constants in the header file are identical to the context strings in the help file, you can include the header file in the [MAP] section to assign context identifiers, as shown in the following example:

```
[MAP]
#include <myhelp.h>
```

If a help file contains two or more Contents topics, the application can assign one as the default by using the context identifier and the HELP\_SETCONTENTS value in a call to the WinHelp function.

### Choosing Help with the Keyboard

An application can enable the user to choose a help topic with the keyboard by intercepting the F1 key. Intercepting this key lets the user select a menu, menu item, dialog box, message box, or control window and view Help for it with a single keystroke.

To intercept the F1 key, the application must install a message-filter procedure by using the SetWindowsHook function. This allows the application to examine all keystrokes for the application, regardless of which window has the input focus. If the filter procedure detects the F1 key, it posts a WM\_F1DOWN message (application-defined) to the application's main window procedure. The procedure then determines which help topic to display.

The filter procedure should have the following form:

```
int FAR PASCAL FilterFunc(nCode, wParam, lParam)
int nCode;
WORD wParam;
DWORD lParam;
{
    LPMSG lpmsg = (LPMSG)lParam;

    if ((nCode == MSGF_DIALOGBOX || nCode == MSGF_MENU) &&
        lpmsg->message == WM_KEYDOWN && lpmsg->wParam == VK_F1) {
        PostMessage(hWnd, WM_F1DOWN, nCode, 0L);
    }

    DefHookProc(nCode, wParam, lParam, &lpFilterFunc);

    return 0;
}
```

The application should install the filter procedure after creating the main window, as shown in the following example:

```
lpProcInstance = MakeProcInstance(FilterFunc, hInstance);
if (lpProcInstance == NULL)
    return FALSE;

lpFilterFunc = SetWindowsHook(WH_MSGFILTER, lpProcInstance);
```

Like all callback functions, the filter procedure must be exported by the application.

The filter procedure sends a WM\_F1DOWN message only when the F1 key is pressed in a dialog box, message box, or menu. Many applications also display the Contents topic if no menu, dialog box, or message box is selected when the user presses the F1 key. In this case, the application should define the F1 key as an accelerator key that starts Help.

To create an accelerator key, the application's resource-definition file must define an accelerator table, as follows:

```
1 ACCELERATORS
BEGIN
    VK_F1, IDM_HELP_CONTENTS, VIRTKEY
END
```

To support the accelerator key, the application must load the accelerator table by using the LoadAccelerators function and translate the accelerator keys in the main message loop by using the TranslateAccelerator function.

In addition to installing the filter procedure, the application must keep track of which menu, menu item, dialog box, or message box is currently selected. In other words, when the user selects an item, the application must set a global variable indicating the current context. For dialog and message boxes, the application should set the global variable immediately before calling the DialogBox or MessageBox function. For menus and menu items, the application should set the variable whenever it receives a WM\_MENUSELECT message. As long as identifiers for all menu items and controls in an application are unique, an application can use code similar to the following example to monitor menu selections:

```
case WM_MENUSELECT:
    /*
    * Set dwCurrentHelpId to the Help ID of the menu item that is
```

```

* currently selected.
*/

    if (HIWORD(lParam) == 0) /* no menu selected */
        dwCurrentHelpId = ID_NONE;

    else if (lParam & MF_POPUP) { /* pop-up selected*/
        if ((HMENU)wParam == hMenuFile)
            dwCurrentHelpId = ID_FILE;
        else if ((HMENU)wParam == hMenuEdit)
            dwCurrentHelpId = ID_EDIT;
        else if ((HMENU)wParam == hMenuHelp)
            dwCurrentHelpId = ID_HELP;
        else
            dwCurrentHelpId = ID_SYSTEM;
    }

    else /* menu item selected */
        dwCurrentHelpId = wParam;

    break;

```

In this example, the `hMenuFile`, `hMenuEdit`, and `hMenuHelp` parameters must previously have been set to specify the corresponding menu handles. An application can use the [GetMenu](#) and [GetSubMenu](#) functions to retrieve these handles.

When the main window procedure finally receives a `WM_F1DOWN` message, it should use the current value of the global variable to display a help topic. The application can also provide Help for individual controls in a dialog box by determining which control has the focus at this point, as shown in the following example:

```

case WM_F1DOWN:
    /*
    * If there is a current Help context, display it.
    */

    if (dwCurrentHelpId != ID_NONE) {
        DWORD dwHelp = dwCurrentHelpId;

        /*
        * Check for context-sensitive Help for individual dialog
        * box controls.
        */

        if (wParam == MSGF_DIALOGBOX) {
            WORD wID = GetWindowWord(GetFocus(), GWW_ID);
            if (wID != IDOK && wID != IDCANCEL)
                dwHelp = (DWORD) wID;
        }

        WinHelp(hWnd, szHelpFileName, HELP_CONTEXT, dwHelp);

        /*
        * This call is used to remove the highlighting from the
        * System menu, if necessary.
        */

        DrawMenuBar(hWnd);
    }

    break;

```

When the application ends, it must remove the filter procedure by using the [UnhookWindowsHook](#) function and free the procedure instance for the function by using the [FreeProcInstance](#) function.

### Choosing Help with the Mouse

An application can enable the user to choose a help topic with the mouse by intercepting mouse input messages and calling the `WinHelp` function. To distinguish requests to view Help from regular mouse input, the user must press the `SHIFT+F1` key combination. In such cases, the application sets a global variable when the user presses the key combination and changes the cursor shape to a question-mark pointer to indicate that the mouse can be used to choose a help topic.

To detect the `SHIFT+F1` key combination, an application checks for the `VK_F1` virtual-key value in each `WM_KEYDOWN` message sent to its main window procedure. It also checks for the `VK_ESCAPE` virtual-key code. The user presses the `ESC` key to quit Help and restore the mouse to its regular function. The following example checks for these keys:

```
case WM_KEYDOWN:
    if (wParam == VK_F1) {

        /* If Shift-F1, turn Help mode on and set Help cursor. */

        if (GetKeyState(VK_SHIFT)) {
            bHelp = TRUE;
            SetCursor(hHelpCursor);
            return DefWindowProc(hwnd, message, wParam, lParam);
        }

        /* If F1 without shift, call Help main index topic. */

        else {
            WinHelp(hwnd, "myhelp.hlp", HELP_CONTENTS, 0L);
        }

        else if (wParam == VK_ESCAPE && bHelp) {

            /* Escape during Help mode: turn Help mode off. */

            bHelp = FALSE;
            SetCursor((HCURSOR) GetClassWord(hwnd, GCW_HCURSOR));
        }

        break;
    }
```

Until the user clicks the mouse or presses the `ESC` key, the application responds to `WM_SETCURSOR` messages by resetting the cursor to the arrow and question-mark combination.

```
case WM_SETCURSOR:
    /*
    * In Help mode, it is necessary to reset the cursor in response
    * to every WM_SETCURSOR message. Otherwise, by default, Windows
    * will reset the cursor to that of the window class.
    */

    if (bHelp) {
        SetCursor(hHelpCursor);
        break;
    }

    return (DefWindowProc(hwnd, message, wParam, lParam));

case WM_INITMENU:
    if (bHelp) {
        SetCursor(hHelpCursor);
    }

    return (TRUE);
```

If the user clicks the mouse button in a nonclient area of the application window while in Help mode, the application receives a `WM_NCLBUTTONDOWN` message. By examining the `wParam` value of this message, the application can determine which context identifier to pass to `WinHelp`.

```

case WM_NCLBUTTONDOWN:
    /*
    * If in Help mode (Shift+F1), display context-sensitive
    * Help for nonclient area.
    */

    if (bHelp) {
        dwHelpContextId =
        (wParam == HTCAPTION) ?(DWORD) HELPID_TITLE_BAR:
        (wParam == HTSIZE) ? (DWORD) HELPID_SIZE_BOX:
        (wParam == HTREDUCE) ? (DWORD) HELPID_MINIMIZE_ICON:
        (wParam == HTZOOM) ? (DWORD) HELPID_MAXIMIZE_ICON:
        (wParam == HTSYSTEMMENU) ?(DWORD) HELPID_SYSTEM_MENU:
        (wParam == HTBOTTOM) ? (DWORD) HELPID_SIZING_BORDER:
        (wParam == HTBOTTOMLEFT) ? (DWORD) HELPID_SIZING_BORDER:
        (wParam == HTBOTTOMRIGHT) ?(DWORD) HELPID_SIZING_BORDER:
        (wParam == HTTOP) ?(DWORD) HELPID_SIZING_BORDER:
        (wParam == HTLEFT) ?(DWORD) HELPID_SIZING_BORDER:
        (wParam == HTRIGHT) ?(DWORD) HELPID_SIZING_BORDER:
        (wParam == HTTOPLEFT) ?(DWORD) HELPID_SIZING_BORDER:
        (wParam == HTTOPRIGHT) ? (DWORD) HELPID_SIZING_BORDER:
        (DWORD) 0L;

        if (!(BOOL) dwHelpContextId)
        return DefWindowProc(hwnd, message, wParam, lParam);
        bHelp = FALSE;
        WinHelp(hwnd, szHelpFileName, HELP_CONTEXT, dwHelpContextId);
        break;
    }

    return (DefWindowProc(hwnd, message, wParam, lParam));

```

If the user clicks a menu item while in Help mode, the application intercepts the WM\_COMMAND message and sends the Help request:

```

case WM_COMMAND:

    /* In Help mode (Shift-F1)? */

    if (bHelp) {
        bHelp = FALSE;
        WinHelp(hwnd, szHelpFileName, HELP_CONTEXT, (DWORD)wParam);
        return NULL;
    }

```

### Searching for Help with Keywords

An application can enable the user to search for help topics based on full or partial keywords. This method is similar to employing the Search dialog box in Windows Help to find useful topics. The following example searches for the keyword "Keyboard" and displays the corresponding topic, if found:

```
WinHelp(hwnd, "myhelp.hlp", HELP_KEY, "Keyboard");
```

If the topic is not found, Windows Help displays an error message. If more than one topic has the same keyword, Windows Help displays only the first topic.

An application can give the user more options in a search by specifying partial keywords. When a partial keyword is given, Windows Help usually displays the Search dialog box to allow the user to continue the search or return to the application. However, if there is an exact match and no other topic exists with the given keyword, Windows Help displays the topic. The following example opens the Search dialog box and selects the first keyword in the list starting with the letters Ke:

```
WinHelp(hwnd, "myhelp.hlp", HELP_PARTIALKEY, "Ke");
```

When the HELP\_KEY and HELP\_PARTIALKEY values are specified in the WinHelp function, Windows Help searches the K keyword table. This table contains keywords generated by using the letter K with \

footnote statements in the topic file. An application can search alternative keyword tables by specifying the `HELP_MULTIKEY` value in the `WinHelp` function. In this case, the application must specify the footnote character and the full keyword in a MULTIKEYHELP structure, as follows:

```
HGLOBAL hglblmkh;
MULTIKEYHELP FAR* mkh;
PSTR pszKeyword = "Frame";
UINT cb;

cb = sizeof(MULTIKEYHELP) + strlen(pszKeyword);

hglblmkh = GlobalAlloc(GHND, (DWORD) cb);
if (hglblmkh == NULL)
    break;
mkh = (MULTIKEYHELP FAR*) GlobalLock(hglblmkh);

mkh->mkSize    = cb;
mkh->mkKeylist = 'L';
lstrcpy(mkh->szKeyphrase, pszKeyword);

WinHelp(hwnd, "myhelp.hlp", HELP_MULTIKEY, (DWORD) mkh);

GlobalUnlock(hglblmkh);
GlobalFree(hglblmkh);
```

If the keyword is not found, Windows Help displays an error message. If more than one topic has the keyword, Windows Help displays only the first topic.

Applications cannot use alternative keyword tables unless the `MULTIKEY` option is specified in the [OPTIONS] section of the project file.

#### Displaying Help in a Secondary Window

An application can display help topics in secondary windows instead of in Windows Help's main window. Secondary windows are useful whenever the user does not need the full capabilities of Windows Help. The Windows Help menus and buttons are not available in secondary windows.

To display Help in a secondary window, the application specifies the name of the secondary window along with the name of the help file. The following example displays the help topic having the context identifier `IDM_FILE_SAVE` in the secondary window named `wnd_menu`:

```
WinHelp(hwnd, "myhelp.hlp>wnd_menu", HELP_CONTEXT, IDM_FILE_SAVE);
```

The name and characteristics of the secondary window must be defined in the [WINDOWS] section of the project file, as in the following example:

```
[WINDOWS]
wnd_menu = "Menus", (128, 128, 256, 256), 0
```

Windows Help displays the secondary window with the initial size and position specified in the [WINDOWS] section. However, an application can set a new size and position by specifying the HELP\_SETWINPOS value in the `WinHelp` function. In this case, the application sets the members in a HELPWININFO structure to specify the window size and position. The following examples sets the secondary window `wnd_menu` to a new size and position:

```
HANDLE hhwi;
LPHELPWININFO lphwi;
WORD wSize;
char *szWndName = "wnd_menu";

wSize = sizeof(HELPWININFO) + strlen(szWndName);
hhwi = GlobalAlloc(GHND, wSize);
lphwi = (LPHELPWININFO)GlobalLock(hhwi);

lphwi->wStructSize = wSize;
lphwi->x = 256;
lphwi->y = 256;
lphwi->dx = 767;
```



```
lphwi->dy= 512;  
lphwi->wMax = 0;  
lstrcpy(lphwi->rgchMember, szWndName);  
  
WinHelp(hwnd, "myhelp.hlp", HELP_SETWINPOS, lphwi);  
  
GlobalUnlock(hhwi);  
GlobalFree(hhwi);
```

#### Canceling Help

Windows Help requires an application to explicitly cancel Help so that Windows Help can free any resources it used to keep track of the application and its help files. The application can do this at any time.

An application cancels Windows Help by calling the WinHelp function and specifying the `HELP_QUIT` value, as shown in the following example:

```
WinHelp(hwnd, "myhelp.hlp", HELP_QUIT, NULL);
```

If the application has made any calls to the WinHelp function, it must cancel Help before it closes its main window (for example, in response to the WM\_DESTROY message in the main window procedure). An application needs to call WinHelp only once to cancel Help, no matter how many help files it has opened. Windows Help remains running until all applications or dynamic-link libraries that have called the WinHelp function have canceled Help.

## CODE Module Definition Statement

### CODE Module-Definition Statement

CODE attributes `[[FIXED|MOVEABLE]] [[DISCARDABLE]] [[\PRELOAD|LOADONCALL]]`

The CODE statement specifies the attributes of code segments.

#### Parameters

This statement takes no parameters. However, options selected from the following list must be specified:

| Option       | Meaning                                                                                                      |
|--------------|--------------------------------------------------------------------------------------------------------------|
| <u>FIXED</u> | Specifies that the segment remains at a fixed memory location.                                               |
| MOVEABLE     | Specifies that the segment can be moved, if necessary, in order to compact memory.                           |
| DISCARDABLE  | Specifies that the segment can be discarded if it is no longer needed.                                       |
| PRELOAD      | Specifies that the segment is loaded when the module is first loaded.                                        |
| LOADONCALL   | Specifies that the segment is loaded when it is called. The Resource Compiler (RC) may override this option. |

#### Comments

There are no default attributes for code segments. The .DEF file should always define code-segment attributes explicitly.

The FIXED and MOVEABLE options are mutually exclusive. The PRELOAD and LOADONCALL options are mutually exclusive: If options conflict with each other, MOVEABLE overrides FIXED and PRELOAD overrides LOADONCALL.

#### Example

The following example sets defaults for the module's code segments so that they are movable and are not loaded until accessed.

```
CODE MOVEABLE LOADONCALL
```

## DATA Module Definition Statement

### DATA Module-Definition Statement

DATA [[NONE|SINGLE|MULTIPLE]] [[FIXED|MOVEABLE]]

The DATA statement specifies the attributes of the standard data segment, which is all application segments belonging to the DGROUP group and the DATA class. In C applications, the standard data segment is created automatically. The data is always preloaded.

#### Parameters

This statement takes no parameters. However, options selected from the following list must be specified:

| Option       | Meaning                                                                                                                                                     |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NONE         | Specifies that there is no data segment. To be effective, this option should be the only attribute of the segment. This option is valid only for libraries. |
| SINGLE       | Specifies that a single data segment is shared by all instances of the module. This option is valid only for libraries.                                     |
| MULTIPLE     | Specifies that one data segment exists for each instance. This option is valid only for applications.                                                       |
| PRELOAD      | Specifies that the segment is loaded when the module is first loaded.                                                                                       |
| <u>FIXED</u> | Specifies that the segment remains at a fixed memory location.                                                                                              |
| MOVEABLE     | Specifies that the segment can be moved, if necessary, in order to compact memory.                                                                          |

#### Comments

There are no default attributes for data segments. The .DEF file should always define data-segment attributes explicitly. Data segments are always preloaded.

The NONE, SINGLE, and MULTIPLE options are mutually exclusive.

The FIXED and MOVEABLE options are mutually exclusive.

If options conflict with each other, MULTIPLE overrides NONE, SINGLE overrides NONE, and MOVEABLE overrides FIXED.

#### Example

The following example defines application's data segment so that it can be moved. It also specifies that a single data segment is shared by all instances of the module.

```
DATA MOVEABLE SINGLE
```

## DESCRIPTION Module Definition Statement

### DESCRIPTION Module-Definition Statement

DESCRIPTION `text`

The DESCRIPTION statement inserts text into the application module. It is useful for embedding version-control or copyright information.

| Parameter | Description                                                     |
|-----------|-----------------------------------------------------------------|
| text      | Specifies a one-line string enclosed in single quotation marks. |

#### Example

The following example embeds the text "Microsoft Windows Template Application" in the application module.

```
DESCRIPTION 'Microsoft Windows Template Application'
```

## EXETYPE Module Definition Statement

EXETYPE Module-Definition Statement EXETYPE headertype

The EXETYPE statement specifies the default executable-file (.EXE) header type. The statement is required for every Windows application.

| Parameter  | Description                                                                                                                             |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| headertype | Specifies the header type. When linking an application intended for the Windows environment, set this parameter to the value "WINDOWS". |

### Example

The following example specifies Windows as the .EXE header type.

```
EXETYPE WINDOWS
```

## EXPORTS Module Definition Statement

### EXPORTS Module-Definition Statement

EXPORTS exportname [[ordinal-option]] [[\res-option]] [[data-option]] [[parameter-option]]

The EXPORTS statement specifies the names and attributes of the functions to be exported to other applications. The EXPORTS keyword marks the beginning of the definitions. It can be followed by any number of export definitions, each on a separate line.

| Parameter        | Description                                                                                                                                                                                                                                                                                                                                                                                             |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| exportname       | Specifies the name of the function to be exported. This name consists of one or more ASCII characters in the following format:<br><br><entryname>=[[internalname]]<br>The entryname parameter specifies the name to be used by other applications to access the exported function, and internalname, an optional parameter specifies the actual name of the function if entryname is not its real name. |
| ordinal-option   | Defines ordinal value of the function. This parameter is an integer and has the following format:<br><br>@ordinal<br>The ordinal value defines the location of the function's name in the application's string table.                                                                                                                                                                                   |
| res-option       | Specifies the optional keyword RESIDENTNAME, which stipulates that the function's name must be resident at all times.                                                                                                                                                                                                                                                                                   |
| data-option      | Specifies the optional keyword NODATA, which stipulates that the function is not bound to a specific data segment. When called, the function uses the current data segment.                                                                                                                                                                                                                             |
| parameter-option | Specifies an integer value for the number of words the function expects to be passed as parameters. This parameter is optional.                                                                                                                                                                                                                                                                         |

#### Comments

When exporting functions from libraries, use an ordinal value rather than a name; using an ordinal conserves space.

Functions using the C calling convention (declared with the `_cdecl` keyword) must be exported with a leading underscore. For example, the following statement exports the `MyPrintf` function:

```
EXPORTS
    _MyPrintf
```

In addition, applications calling this function must explicitly import the function by declaring it (with the leading underscore) in the IMPORTS section of the application's module-definition (`.DEF`) file.

Functions using the `_fastcall` calling convention must be explicitly exported and imported with a leading `@` symbol.

#### Example

The following example exports the `SampleRead`, `StringIn` and `CharTest` functions so that other applications, or Windows itself, can call them.

```
EXPORTS
    SampleRead=read2bin @1 8
    StringIn=str1 @2 4
    CharTest NODATA
```

## HEAPSIZE Module Definition Statement

HEAPSIZE Module-Definition Statement HEAPSIZE bytes

The HEAPSIZE statement specifies the number of bytes needed by the application for its local heap. An application uses the local heap whenever it allocates local memory. The size of the local heap must be at least large enough to hold the current environment for an application.

| Parameter | Description                                                                                                                                                                         |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bytes     | Specifies the heap size in bytes. The default heap size is zero; the minimum size is 256 bytes. The heap size must not exceed 65,536 bytes (the size of a single physical segment). |

### Example

This example sets the size of the application's local heap to 4,096 bytes.

```
HEAPSIZE 4096
```

## IMPORTS Module Definition Statement

### IMPORTS Module-Definition Statement

IMPORTS [[internal-option]] modulename [[entry-option]]

The IMPORTS statement specifies the names and attributes of the functions to be imported from dynamic-link libraries (DLLs). The IMPORTS keyword marks the beginning of the definitions. It can be followed by any number of import definitions, each on a separate line.

| Parameter       | Description                                                                                                                                                                                                                                                                                                                            |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| internal-option | Specifies the name of the function to be imported. This name consists of one or more ASCII characters in the following format:<br><br>internal-name=<br>The internal-name parameter specifies the name to be used by the application to call the function. This name must be unique.                                                   |
| modulename      | Specifies one or more ASCII characters that constitute the name of the executable module containing the function. The module name must match the name of the executable file. For example, an application with the executable file SAMPLE.DLL has the module name "SAMPLE". The executable file must be named with the .DLL extension. |
| entry-option    | Specifies the function to be imported. This parameter can be either .entryname or .entryordinal, where entryname is the actual name of the function and entryordinal is the ordinal value of the function.                                                                                                                             |

### Comments

Instead of listing imported DLL functions in the IMPORTS statement, you can specify an "import library" for the DLL in the LINK command line for your application.

Functions using the \_cdecl or \_fastcall calling conventions, however, must be explicitly imported in the module-definition file for the application (using either a leading underscore or a leading @ symbol, respectively).

### Example

```
IMPORTS
    Sample.SampleRead
    write2hex=Sample.SampleWrite
    Read.1
```



## LIBRARY Module Definition Statement

LIBRARY Module-Definition Statement LIBRARY libraryname

The LIBRARY statement specifies the name of a library module. Library modules are resource modules that contain code, data, and other resources but are not executed as independent programs.

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                                          |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| libraryname | Specifies one or more ASCII characters that constitute the name of the library module. A library's module name must match the name of the executable file. For example, the library USER.EXE has the module name "USER".<br>The libraryname parameter is optional. If it is not included, LINK takes the library name from the filename (without extension) for the executable file. |

### Comments

The starting address of the library module is determined by the object files for the library; it is an internally defined function.

If the .DEF file includes neither a NAME nor a LIBRARY statement, LINK uses a NAME statement without a modulename parameter as the default.

### Example

This example gives a library the module name "Utilities."

```
LIBRARY Utilities
```

## NAME Module Definition Statement

NAME Module-Definition Statement NAME modulename

The NAME statement specifies the name of the executable module for the application. The module name identifies the module when exporting functions.

| Parameter  | Description                                                                                                                                                                                                                                                                                                                                                                 |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| modulename | Specifies one or more uppercase ASCII characters that constitute the name of the executable module. The module name must match the name of the executable file. For example, an application with the executable file SAMPLE.EXE has the module name "SAMPLE". Do not use system library names; examples of these names are KERNEL, USER, GDI, SHELL, COMMDBG, and TOOLHELP. |

The modulename parameter is optional. If it is not included, LINK takes the module name from the filename (without extension) of the executable file. For example, if you do not specify a module name and the executable file is named MYAPP.EXE, LINK assumes that the module name is "MYAPP".

### Comments

If the .DEF file includes neither a NAME nor a LIBRARY statement, LINK uses a NAME statement without a modulename parameter as the default.

### Example

This example gives an application the module name "Calendar".

```
NAME Calendar
```

## SEGMENTS Module Definition Statement

### SEGMENTS Module-Definition Statement

SEGMENTS segmentname [[CLASS 'class-name']] [[minalloc]] \ [[FIXED|MOVEABLE]] [ [DISCARDABLE]] [[SHARED|NONSHARED]] [[PRELOAD|LOADONCALL]]

The SEGMENTS statement specifies the segment attributes of additional code and data segments.

#### Parameters

This statement takes no parameters. However, options selected from the following list must be specified:

| Option      | Meaning                                                                                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FIXED       | Specifies that the segment remains at a fixed memory location.                                                                                                              |
| MOVEABLE    | Specifies that the segment can be moved if necessary, in order to compact memory.                                                                                           |
| DISCARDABLE | Specifies that the segment can be discarded if it is no longer needed.                                                                                                      |
| PRELOAD     | Specifies that the segment is loaded when the module is first loaded.                                                                                                       |
| LOADONCALL  | Specifies that the segment is loaded when it is accessed or called. The Resource Compiler (RC) may override this option. For more information, see Microsoft Windows Tools. |

| Parameter   | Description                                                                                                                                                                                                                                               |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| segmentname | Specifies one or more ASCII characters that constitute the name of the new segment. This parameter can be any name, including the standard segment names <code>_TEXT</code> and <code>_DATA</code> , which represent the standard code and data segments. |
| class-name  | Specifies the class name of the segment. If no class name is specified, LINK uses the <code>CODE</code> class name by default.                                                                                                                            |
| minalloc    | Specifies the minimum allocation size for the segment. This value must be an integer. The <code>minalloc</code> parameter is optional.                                                                                                                    |

#### Comments

There are no default attributes for additional segments. The `.DEF` file should always define the attributes of additional segments explicitly.

The `FIXED` and `MOVEABLE` options are mutually exclusive. The `PRELOAD` and `LOADONCALL` options are mutually exclusive. If options conflict with each other, `MOVEABLE` overrides `FIXED` and `PRELOAD` overrides `LOADONCALL`.

#### Example

The following example defines the segment named `_TEXT` as `FIXED`. It specifies the `_INIT` segment as `PRELOAD` and `DISCARDABLE`. The `_RES` segment of the `data` class becomes `PRELOAD` and `DISCARDABLE`.

```
SEGMENTS
  _TEXT FIXED
  _INIT PRELOAD DISCARDABLE
  _RES CLASS 'DATA' PRELOAD DISCARDABLE
```

## STACKSIZE Module Definition Statement

STACKSIZE Module-Definition Statement STACKSIZE bytes

The STACKSIZE statement specifies the number of bytes needed by the application for its local stack. An application uses the local stack whenever it makes function calls.

| Parameter | Description                                                                                                                                                                                                                                           |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bytes     | Specifies the stack size, in bytes. If the application makes no function calls, the default stack size is zero. If your application does make function calls and you specify a stack size smaller than 5K, Windows automatically sets the size to 5K. |

### Comments

Do not use the STACKSIZE statement for dynamic-link libraries (DLLs).

### Example

This example sets the size of an application's stack to 6,144 bytes.

```
STACKSIZE 6144
```

## STUB Module Definition Statement

STUB Module-Definition Statement `STUB `filename``

The STUB statement appends the old-style executable file specified by filename to the beginning of the module. The executable stub should display a warning message and stop execution if the user attempts to run the module without having loaded Windows. The default file WINSTUB.EXE can be used if no other actions are required.

| Parameter | Description                                                                                                                   |
|-----------|-------------------------------------------------------------------------------------------------------------------------------|
| filename  | Specifies the name of the old-style executable file to be appended to the module. The name must have the DOS filename format. |

### Comments

If the file named by filename is not in the current directory, LINK searches for the file in the directories specified in PATH environment variable.

### Example

This example specifies the executable file WINSTUB.EXE as the stub for the application. If a user tries to run this application in the DOS environment rather than with the Windows operating system, WINSTUB.EXE starts instead.

```
STUB 'WINSTUB.EXE'
```

## Module Definition Statements

CODE Module Definition Statement  
DATA Module Definition Statement  
DESCRIPTION Module Definition Statement  
EXETYPE Module Definition Statement  
EXPORTS Module Definition Statement  
HEAPSIZE Module Definition Statement  
IMPORTS Module Definition Statement  
LIBRARY Module Definition Statement  
NAME Module Definition Statement  
SEGMENTS Module Definition Statement  
STACKSIZE Module Definition Statement  
STUB Module Definition Statement

Defines attributes of standard code segment  
defines attributes of standard data segment  
Inserts text into application module  
Specifies the default .EXE header type  
Specifies functions to export to other apps  
Specifies size of local heap  
Specifies functions to import from DLLs  
Specifies name of a library module  
Specifies name of executable module  
Specifies segment attributes  
Specifies size of local stack  
Appends stub to the beginning of the module

## [ALIAS] Section

```
[ALIAS]
context_string = alias
.
.
.
```

The [ALIAS] section assigns one or more context strings to the same topic alias. This section is optional.

| Parameter      | Description                                                                                                                                                                                                                                                                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| context_string | Specifies the context string that identifies a particular topic. This context string may be used in a hotspot or in the <a href="#">[MAP] section</a> to refer to a particular topic.                                                                                                                                                                                  |
| alias          | Specifies the alternative string or alias name. This string is used in the \footnote statement. An alias string has the same form and follows the same conventions as the topic context string. That is, it is not case-sensitive and may contain the alphabetic characters A through Z, the numeric characters 0 through 9, and the period and underscore characters. |

### Comments

Because context strings must be unique for each topic and cannot be used for any other topic in the Help project, the [ALIAS] section provides a way to delete or combine help topics without recoding your files. For example, if you create a topic that replaces information in three other topics, you could manually search through your files for invalid cross-references to the deleted topics. The easier approach, however, would be to use the [ALIAS] section to assign the name of the new topic to the deleted topics.

The [ALIAS] section can also be used when your application has multiple context identifiers for one help topic. This situation occurs in context-sensitive Help.

Alias names can be used in a [\[MAP\]](#) section, but only if the [ALIAS] section precedes the [MAP] section.

### Example

The following example creates several aliases:

```
[ALIAS]
sm_key=key_shrtcuts
cc_key=key_shrtcuts
st_key=key_shrtcuts; combined into Keyboard Shortcuts topic
clskey=us_dlog_bxs
maakey=us_dlog_bxs ; covered in Using Dialog Boxes topic.
chk_key=dlogprts
drp_key=dlogprts
lst_key=dlogprts
opt_key=dlogprts
tbx_key=dlogprts ; combined into Parts of Dialog Box topic.
frmtxt=edittxt
wrptxt=edittxt
seltxt=edittxt; covered in Editing Text topic.
```

### See Also

[\[MAP\]](#)

## [BAGGAGE] Section

[BAGGAGE]

filename

.

.

.

The [BAGGAGE] section lists files (typically multimedia elements) that the Microsoft Help Compiler stores within the help file's internal file system. Windows Help can access data files stored in the help file more efficiently than it can access files in the normal MS-DOS file system, since it doesn't have to read the file allocation table from CD-ROM.

| Parameter | Description                                                                                  |
|-----------|----------------------------------------------------------------------------------------------|
| filename  | Specifies the full path of a file. If a file cannot be found, the compiler reports an error. |

### Comments

A maximum of 1,000 files can be stored as baggage files.

If a file is listed in the [BAGGAGE] section, you must use or write a dynamic-link library that uses Windows Help to read these files from the help file.

See Also

ROOT



■ [BITMAPS] Section

[BITMAPS]  
filename

.  
.  
.

The [BITMAPS] section specifies the names and locations of the bitmap files specified in the bmc, bml, and bmr statements.

| Parameter | Description                                                                                         |
|-----------|-----------------------------------------------------------------------------------------------------|
| filename  | Specifies the full path of a bitmap file. If a file cannot be found, the compiler reports an error. |

Comments

For Windows 3.1, the [BITMAPS] section is not required if the bitmaps are located in the Help project directory or if the path containing the bitmaps is listed in the BMROOT or ROOT option. If the project file does not include either of these options, each bitmap filename must be listed in the [BITMAPS] section of the project file.

Example

The following example specifies three bitmap files:

```
[ BITMAPS ]  
BMP01 . BMP  
BMP02 . BMP  
BMP03 . BMP
```

See Also

BMROOT, ROOT

### Changes for Windows 3.1

For Windows 3.1, the [BITMAPS] section is not required if the bitmaps are located in the Help project directory or if the path containing the bitmaps is listed in the BMROOT or ROOT option. For Windows 3.0, all bitmaps used in the help file must be placed in the [BITMAPS] section.

## BMROOT Option

BMROOT = path[, path]...

The BMROOT option specifies the directory containing the bitmap files specified in the bmc, bml, and bmr statements.

| Parameter | Description                      |
|-----------|----------------------------------|
| path      | Specifies a drive and full path. |

### Comments

If the project file has a BMROOT option, you do not need to list the bitmap files in the [BITMAPS] section.

If the project file does not have a BMROOT option, the Help compiler looks for bitmaps in the directories specified by the ROOT option. If the project file does not have a ROOT option or if the ROOT option does not specify the directory containing the bitmap files, the filename for each bitmap must be specified in the [BITMAPS] section.

### Example

The following example specifies that bitmaps are in the \HELP\BMP directory on drive C: and the \GRAPHICS\ART directory on drive D:

```
[ OPTIONS ]  
BMROOT=C:\HELP\BMP , D:\GRAPHICS\ART
```

### See Also

[BITMAPS], [OPTIONS], ROOT

## BUILD Option

`BUILD = expression`

The BUILD option specifies which topics containing build tags are included in a build. The BUILD option does not apply to topics that do not contain build tags.

A topic contains a build tag if it contains a build-tag \footnote statement. Topics without build tags are always compiled, regardless of the current build expression.

| Parameter  | Description                                                                                                                                                                                                                   |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| expression | Specifies the build expression. This parameter consists of a combination of build tags (specified in the <a href="#">[BUILDTAGS]</a> section) and the following operators:                                                    |
| Operator   | Description                                                                                                                                                                                                                   |
| ~          | Applies the NOT operator to a single tag. The Help compiler compiles a topic only if the tag is not present. This operator has the highest precedence; the compiler applies it before any other operator.                     |
| &          | Combines two tags by using the AND operator. The Help compiler compiles a topic only if it contains both tags. The compiler applies this operator only after the ~ operator has been applied.                                 |
|            | Combines two tags by using the OR operator. The Help compiler compiles a topic if it has at least one tag. This operator has the lowest precedence; the compiler applies it only after all other operators have been applied. |
|            | Parentheses may be used to override operator precedence. Expressions enclosed in parentheses are always evaluated first.                                                                                                      |

### Comments

Only one BUILD option can be given per project file.

The Help compiler evaluates all build expressions from left to right, using the specified precedence rules.

### Example

The following examples assume that the [\[BUILDTAGS\]](#) section in the project file defines the build tags DEMO, MASTER, and TEST\_BUILD. Although the following examples show several BUILD options on consecutive lines, only one BUILD option per project file is allowed.

```
BUILD = DEMO ; compile topics that have the DEMO tag
BUILD = DEMO & MASTER ; compile topics with both DEMO and MASTER
BUILD = DEMO | MASTER ; compile topics with either DEMO or MASTER
BUILD = ~DEMO ; compile topics that do not have DEMO
BUILD = (DEMO | MASTER) & TEST_BUILD
; compile topics that have TEST_BUILD and
; either DEMO or MASTER
```

### See Also

[\[BUILDTAGS\]](#), [\[OPTIONS\]](#)

## [BUILDTAGS] Section

[BUILDTAGS]

tag

.  
.  
.

The [BUILDTAGS] section defines the build tags for the help file. The Help compiler uses these tags to determine which topics to include when building the help file.

This section is used in conjunction with the build-tag \footnote statements. These \footnote statements associate a build tag with a given topic. If the build tag is also defined in the [BUILDTAGS] section, the Help compiler compiles the topic; otherwise, it ignores the topic.

| Parameter | Description                                                                                                                                                                                                                                                        |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tag       | Specifies a build tag consisting of any combination of characters except spaces. The Help compiler strips any space characters from the tag. Also, the compiler treats uppercase and lowercase characters as the same characters (that is, it is case-insensitive) |

Comments

The [BUILDTAGS] section is optional. If given, it can contain up to 30 build tags.

Example

The following example shows the form of the [BUILDTAGS] section in a sample project file:

```
[BUILDTAGS]
DEMO ; topics to include in demo build
MASTER; topics to include in master help file
DEBUGBUILD ; topics to include in debugging build
TESTBUILD ; topics to include in a mini-build for testing
```

See Also

BUILD

## CITATION Option

CITATION = citation

The CITATION option places a custom citation in the About dialog box of Windows Help. Windows Help displays the citation immediately below the Microsoft copyright notice.

| Parameter | Description                                                                                                                        |
|-----------|------------------------------------------------------------------------------------------------------------------------------------|
| citation  | Specifies the citation. The notice can be any combination of characters; its length must be in the range 35 through 75 characters. |

See Also

COPYRIGHT, [OPTIONS]

## COMPRESS Option

COMPRESS = compression-level

The COMPRESS option specifies the level of compression to be used when building the help file. Compression levels indicate either no compression, medium compression (approximately 40%), or high compression (approximately 50%).

| Parameter         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |   |                |   |                  |       |                |      |                  |        |                    |    |                |      |                  |     |                  |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---|----------------|---|------------------|-------|----------------|------|------------------|--------|--------------------|----|----------------|------|------------------|-----|------------------|
| compression-level | Specifies the level of compression. This parameter can be one of the following values:                                                                                                                                                                                                                                                                                                                                                                             |       |         |   |                |   |                  |       |                |      |                  |        |                    |    |                |      |                  |     |                  |
|                   | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>No compression</td></tr><tr><td>1</td><td>High compression</td></tr><tr><td>FALSE</td><td>No compression</td></tr><tr><td>HIGH</td><td>High compression</td></tr><tr><td>MEDIUM</td><td>Medium compression</td></tr><tr><td>NO</td><td>No compression</td></tr><tr><td>TRUE</td><td>High compression</td></tr><tr><td>YES</td><td>High compression</td></tr></tbody></table> | Value | Meaning | 0 | No compression | 1 | High compression | FALSE | No compression | HIGH | High compression | MEDIUM | Medium compression | NO | No compression | TRUE | High compression | YES | High compression |
| Value             | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |   |                |   |                  |       |                |      |                  |        |                    |    |                |      |                  |     |                  |
| 0                 | No compression                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |   |                |   |                  |       |                |      |                  |        |                    |    |                |      |                  |     |                  |
| 1                 | High compression                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |   |                |   |                  |       |                |      |                  |        |                    |    |                |      |                  |     |                  |
| FALSE             | No compression                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |   |                |   |                  |       |                |      |                  |        |                    |    |                |      |                  |     |                  |
| HIGH              | High compression                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |   |                |   |                  |       |                |      |                  |        |                    |    |                |      |                  |     |                  |
| MEDIUM            | Medium compression                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |   |                |   |                  |       |                |      |                  |        |                    |    |                |      |                  |     |                  |
| NO                | No compression                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |   |                |   |                  |       |                |      |                  |        |                    |    |                |      |                  |     |                  |
| TRUE              | High compression                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |   |                |   |                  |       |                |      |                  |        |                    |    |                |      |                  |     |                  |
| YES               | High compression                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |   |                |   |                  |       |                |      |                  |        |                    |    |                |      |                  |     |                  |

### Comments

Depending on the degree of compression requested, the build uses either block compression or a combination of block and key-phrase compression. Block compression compresses the topic data into predefined units known as blocks. Key-phrase compression combines repeated phrases found within the source file(s). The compiler creates a phrase-table file with the .PH extension if one does not already exist. If the compiler finds a file with the .PH extension, it uses that file for the current compilation. Because the .PH file speeds up the compression process when little text has changed since the last compilation, you might want to keep the phrase file if you compile the same Help file several times with compression. However, you will get maximum compression if you delete the .PH file before starting each build.

See Also

[OPTIONS] section

## [CONFIG] Section

### [CONFIG]

macro

.  
.  
.

The [CONFIG] section contains one or more macros that carry out actions, such as enabling browse buttons and registering dynamic-link library (DLL) functions. Windows Help executes the macros when it opens the help file.

| Parameter | Description                     |
|-----------|---------------------------------|
| macro     | Specifies a Windows Help macro. |

### Comments

The [CONFIG] section may include any number of lines. Each line of the [CONFIG] section may be up to 254 characters long.

### Example

The following example registers a DLL, creates a button, enables the browse buttons, and sets the name of the help file containing information about how to use Help:

```
[CONFIG]
RegisterRoutine("bmp", "HDisplayBmp", "USSS")
RegisterRoutine("bmp", "CopyBmp", "v=USS")
CreateButton("btn_up", "&Up", "JumpContents(`HOME.HLP')")
BrowseButtons()
SetHelpOnFile("APPHELP.HLP")
```



## CONTENTS Option

CONTENTS = context-string

The CONTENTS option identifies the context string of the highest-level or Contents topic. This topic is usually a table of contents or index within the help file. Windows Help displays the Contents topic whenever the user clicks the Contents button.

| Parameter      | Description                                                                                                                                                                                                                  |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| context-string | Specifies the context string of a topic in the help file. The string can be any combination of characters, except spaces, and must also be specified in a context-string \footnote statement in some topic in the help file. |

### Comments

If the [OPTIONS] section does not include a CONTENTS option, the compiler assumes that the Contents topic is the first topic encountered in the first listed topic file in the [FILES] section of the project file.

The CONTENTS option is equivalent to the INDEX option that was available in Windows version 3.0.

### Example

The following example sets the topic containing the context string "main\_contents" as the Contents topic:

```
CONTENTS=main_contents
```

### See Also

[FILES], [OPTIONS]

## COPYRIGHT Option

COPYRIGHT = copyright-notice

The COPYRIGHT option places a custom copyright notice in the About dialog box of Windows Help. Windows Help displays the notice immediately below the Microsoft copyright notice.

| Parameter        | Description                                                                                                                                |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| copyright-notice | Specifies the copyright notice. The notice can be any combination of characters; its length must be in the range 35 through 75 characters. |

### Comments

The copyright notice is also appended to topics that are copied to the clipboard, unless it is replaced by using the CITATION option.

### See Also

CITATION, [OPTIONS]

## ERRORLOG Option

ERRORLOG = error-filename

The ERRORLOG option directs the Help compiler to write all error messages to the specified file. The compiler also displays the error messages on the screen.

| Parameter      | Description                                                                                                                                                                                          |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| error-filename | Specifies the name of the file to receive the error messages. This parameter can be a full or partial path if the error file should be written to a directory other than the project root directory. |

### Example

The following example writes all errors during the build to the HLPBUGS.TXT file in the Help project root directory.

```
ERRORLOG=HLPBUGS . TXT
```

See Also

[\[OPTIONS\]](#)

## [FILES] Section

[FILES]  
filename

.  
.  
.

The [FILES] section lists all topic files used to build the help file. Every project file requires a [FILES] section.

| Parameter | Description                                                                                                                                                                                                                                            |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename  | Specifies the full or partial path of a topic file. If a partial path is given, the Help compiler uses the directories specified by the <u>ROOT</u> option to construct a full path. If a file cannot be found, the compiler reports an <u>error</u> . |

### Comments

The #include directive can also be used in the [FILES] section to specify the topic files indirectly by designating a file that contains a list of the topic files.

### Example

The following example specifies four topic files:

```
[FILES]
rtftxt\COMMANDS.RTF
rtftxt\HOWTO.RTF
rtftxt\KEYS.RTF
rtftxt\GLOSSARY.RTF
```

The following example uses the #include directive to specify the topic files indirectly. In this case, the file RTFFILES.H must be in the project file (the Help compiler does not use the INCLUDE environment variable to search for files).

```
[FILES]
#include <rtffiles.h>
```

See Also

ROOT

## FORCEFONT Option

FORCEFONT = fontname

The FORCEFONT option forces the specified font to be substituted for all requested fonts. The option is used to create help files that can be viewed on systems that do not have all fonts available.

| Parameter | Description                                                                                                                                                                                                                                                                |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fontname  | Specifies the name of an available font. Font names must be spelled the same as they are in the Fonts dialog box in Control Panel. Font names cannot exceed 20 characters. If an invalid font name is given, the Help compiler uses the MS Sans Serif font as the default. |

See Also  
[\[OPTIONS\]](#)

## ICON Option

ICON = icon-file

The ICON option identifies the icon file to display when the user minimizes Windows Help.

| Parameter | Description                                                                                     |
|-----------|-------------------------------------------------------------------------------------------------|
| icon-file | Specifies the name of the icon file. This file must have the standard Windows icon-file format. |

See Also

[OPTIONS]

## LANGUAGE Option

LANGUAGE = language-name

The LANGUAGE option sets the sorting order for keywords in the Search dialog box.

| Parameter     | Description                                                                                                                                                                           |       |         |              |                                                            |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------------|------------------------------------------------------------|
| language-name | Specifies the language on which to base sorting. This parameter can be the following:                                                                                                 |       |         |              |                                                            |
|               | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>scandinavian</td><td>Sets the sorting order to the Scandinavian-language order.</td></tr></tbody></table> | Value | Meaning | scandinavian | Sets the sorting order to the Scandinavian-language order. |
| Value         | Meaning                                                                                                                                                                               |       |         |              |                                                            |
| scandinavian  | Sets the sorting order to the Scandinavian-language order.                                                                                                                            |       |         |              |                                                            |

### Comments

The default sorting order is the English-language order.

Microsoft Windows Help version 3.1 supports only English and Scandinavian sorting.

See Also

[OPTIONS]

## [MAP] Section

### [MAP]

context-string context-number

·  
·  
·

The [MAP] section associates context strings (or aliases) with context numbers for context-sensitive Help. The context number corresponds to a value the parent application passes to Windows Help in order to display a particular topic. This section is optional.

| Parameter      | Description                                                                                                                                                                                                                                                                                                                                                                       |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| context-string | Specifies the context string of a topic in the help file. The string can be any combination of characters, except spaces, and must also be specified in a context-string \footnote statement in some topic in the help file.                                                                                                                                                      |
| context-number | Specifies the context number to associate with the context string. The number can be in either decimal or standard C hexadecimal format. Only one context number may be assigned to a context string or alias. Assigning the same number to more than one context string generates a compiler error. At least one space must separate the context number from the context string. |

### Comments

You can define the context strings listed in the [MAP] section either in a help topic or in the [\[ALIAS\]](#) section. The compiler generates a warning message if a context string appearing in the [MAP] section is not defined in any of the topic files or in the [\[ALIAS\]](#) section.

If you use an alias name, the [\[ALIAS\]](#) section must precede the [MAP] section in the Help project file.

The [MAP] section supports two additional statements for specifying context strings and their associated context numbers. The first statement has the following form:

```
#define context-string context-number
```

The context-string and context-number parameters are as described in the Parameters section.

The second statement has the following form:

```
#include "filename"
```

The filename parameter, which can be enclosed in either double quotation marks or angle brackets(<>), specifies the name of a file containing one or more #define statements. The file may contain additional #include statements as well, but files may not be nested in this way more than five deep.

### Example

The following example assigns hexadecimal context numbers to the context strings:

```
[MAP]
Edit_Window0x0001
Control_Menu    0x0002
Maximize_Icon  0x0003
Minimize_Icon  0x0004
Split_Bar      0x0005
Scroll_Bar     0x0006
Title_Bar      0x0007
Window_Border  0x0008
```

### See Also

[\[ALIAS\]](#)



## MAPFONTSIZE Option

MAPFONTSIZE = m:p

The MAPFONTSIZE option maps font sizes specified in topic files to different sizes when they are displayed in the Help window. This option is especially useful if there is a significant size difference between the authoring display and the intended user display.

| Parameter | Description                                                                                                                                                                                                                                                                                                     |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| m         | Specifies the size of the source font. This parameter is either a single point size or a range of point sizes. A range of point sizes consists of the low and high point sizes separated by a hyphen (-). If a range is specified, all fonts in the range are changed to the size specified by the p parameter. |
| p         | Specifies the size of the desired font for the help file.                                                                                                                                                                                                                                                       |

### Comments

Although the [OPTIONS] section can contain up to five font ranges, only one font size or range is allowed with each MAPFONTSIZE statement. If more than one MAPFONTSIZE statement is included, the source font size or range specified in subsequent statements cannot overlap previous mappings.

### Example

The following examples illustrate the use of the MAPFONTSIZE option:

```
MAPFONTSIZE=8:12 ; display all 8-pt. fonts as 12-pt.
```

```
MAPFONTSIZE=12-24:16 ; display fonts from 12 to 24 pts. as 16 pts.
```

See Also

[OPTIONS]

## MULTIKEY Option

MULTIKEY = footnote-character

The MULTIKEY option specifies the footnote character to use for an alternative keyword table. This option is intended to be used in conjunction with topic files that contain \footnote statements for alternative keywords.

| Parameter          | Description                                                              |
|--------------------|--------------------------------------------------------------------------|
| footnote-character | Specifies the case-sensitive letter to be used for the keyword footnote. |

### Comments

Since keyword footnotes are case-sensitive, you should limit your keyword-table footnotes to one case, usually uppercase. If an uppercase letter is specified, the compiler will not include footnotes with the lowercase form of the same letter in the keyword table.

You may use any alphanumeric character for a keyword table except K and k, which are reserved for Help's standard keyword table. There is an absolute limit of five keyword tables, including the standard table. However, depending upon system configuration and the structure of your Help system, a practical limit of only two or three tables may be more realistic. If the compiler cannot create an additional keyword table, the additional table is ignored in the build.

### Example

The following example illustrates how to enable the letter L for a keyword-table footnote:

```
MULTIKEY=L
```

See Also

[OPTIONS]

## OLDKEYPHRASE Option

OLDKEYPHRASE = onoff

The OLDKEYPHRASE option specifies whether an existing key-phrase file should be used to build the help file.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |   |                   |   |                       |       |                   |    |                   |     |                   |    |                       |      |                       |     |                       |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---|-------------------|---|-----------------------|-------|-------------------|----|-------------------|-----|-------------------|----|-----------------------|------|-----------------------|-----|-----------------------|
| onoff     | Specifies whether the existing file should be used. This parameter can be one of the following values:                                                                                                                                                                                                                                                                                                                                                                                    |       |         |   |                   |   |                       |       |                   |    |                   |     |                   |    |                       |      |                       |     |                       |
|           | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Recreate the file</td></tr><tr><td>1</td><td>Use the existing file</td></tr><tr><td>FALSE</td><td>Recreate the file</td></tr><tr><td>NO</td><td>Recreate the file</td></tr><tr><td>OFF</td><td>Recreate the file</td></tr><tr><td>ON</td><td>Use the existing file</td></tr><tr><td>TRUE</td><td>Use the existing file</td></tr><tr><td>YES</td><td>Use the existing file</td></tr></tbody></table> | Value | Meaning | 0 | Recreate the file | 1 | Use the existing file | FALSE | Recreate the file | NO | Recreate the file | OFF | Recreate the file | ON | Use the existing file | TRUE | Use the existing file | YES | Use the existing file |
| Value     | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |   |                   |   |                       |       |                   |    |                   |     |                   |    |                       |      |                       |     |                       |
| 0         | Recreate the file                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |   |                   |   |                       |       |                   |    |                   |     |                   |    |                       |      |                       |     |                       |
| 1         | Use the existing file                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |   |                   |   |                       |       |                   |    |                   |     |                   |    |                       |      |                       |     |                       |
| FALSE     | Recreate the file                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |   |                   |   |                       |       |                   |    |                   |     |                   |    |                       |      |                       |     |                       |
| NO        | Recreate the file                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |   |                   |   |                       |       |                   |    |                   |     |                   |    |                       |      |                       |     |                       |
| OFF       | Recreate the file                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |   |                   |   |                       |       |                   |    |                   |     |                   |    |                       |      |                       |     |                       |
| ON        | Use the existing file                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |   |                   |   |                       |       |                   |    |                   |     |                   |    |                       |      |                       |     |                       |
| TRUE      | Use the existing file                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |   |                   |   |                       |       |                   |    |                   |     |                   |    |                       |      |                       |     |                       |
| YES       | Use the existing file                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |   |                   |   |                       |       |                   |    |                   |     |                   |    |                       |      |                       |     |                       |

See Also

[\[OPTIONS\]](#)

## OPTCDROM Option

OPTCDROM = yesvalue

The OPTCDROM option optimizes a help file for display on CD-ROM by aligning topic files on predefined block boundaries.

| Parameter | Description                                                                                                |
|-----------|------------------------------------------------------------------------------------------------------------|
| yesvalue  | Specifies that the file should be optimized for CD-ROM. This parameter can be any of the following values: |

YES  
TRUE  
1  
ON

See Also  
[OPTIONS]

## [OPTIONS] Section

### [OPTIONS]

option

.  
.  
.

The [OPTIONS] section includes options that control how a help file is built and what feedback the build process displays. If this section is included in the project file, it should be the first section listed, so that the options will apply during the entire build process.

| Parameter           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|-------------|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------|--------------|-------------------------------------------------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|------------------------------------------------------------|-----------------|---------------------------------------------------------------------------------------------------------|------------------|----------------------------------------------------------------------------------------------------------------|-----------------|-----------------------------------------------------------------------------------------|------------------|---------------------------------------------------------------------------------------------------------------|-------------|--------------------------------------------------------------------------------------------------------------|-----------------|-----------------------------------------------------------------------------------------|--------------------|----------------------------------------------------------------------------------------|-----------------|-------------------------------------------------------------------|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------------------------------------------------------------|---------------|------------------------------------------------------------|-------------|-------------------------------------------------------------------------------------------|--------------|-----------------------------------------------------------------------------------------|----------------|---------------------------------------------------------------------------------------------|
| option              | Specifies one of the following project-file options:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
|                     | <table><thead><tr><th>Option</th><th>Description</th></tr></thead><tbody><tr><td><u>BMROOT</u></td><td>Specifies the directory containing the bitmap files named in the bmc, bml, and bmr statements in topic files. This option is new for Windows 3.1.</td></tr><tr><td><u>BUILD</u></td><td>Specifies which topics to include in the build.</td></tr><tr><td><u>CITATION</u></td><td>Specifies a string that is appended to topics that are copied from Windows Help instead of the <u>COPYRIGHT</u> string. This option is new for Windows 3.1.</td></tr><tr><td><u>COMPRESS</u></td><td>Specifies the type of compression to use during the build.</td></tr><tr><td><u>CONTENTS</u></td><td>Specifies the context string of the Contents topic for a help file. This option is new for Windows 3.1.</td></tr><tr><td><u>COPYRIGHT</u></td><td>Adds a unique copyright message for the help file to the About dialog box. This option is new for Windows 3.1.</td></tr><tr><td><u>ERRORLOG</u></td><td>Puts compilation errors in a file during the build. This option is new for Windows 3.1.</td></tr><tr><td><u>FORCEFONT</u></td><td>Forces all authored fonts in the topic files to appear in a different font when displayed in the Help window.</td></tr><tr><td><u>ICON</u></td><td>Specifies the icon file to be displayed when the help file is minimized. This option is new for Windows 3.1.</td></tr><tr><td><u>LANGUAGE</u></td><td>Specifies a different sorting order for help files authored in a Scandinavian language.</td></tr><tr><td><u>MAPFONTSIZE</u></td><td>Maps a font size in the topic file to a different font size in the compiled help file.</td></tr><tr><td><u>MULTIKEY</u></td><td>Specifies an alternative keyword table to use for mapping topics.</td></tr><tr><td><u>OLDKEYPHRASE</u></td><td>Specifies whether the compiler should use the existing keyphrase table or create a new one during the build. This option is new for Windows 3.1.</td></tr><tr><td><u>OPTCDROM</u></td><td>Optimizes the help file for CD-ROM use. This option is new for Windows 3.1.</td></tr><tr><td><u>REPORT</u></td><td>Controls the display of messages during the build process.</td></tr><tr><td><u>ROOT</u></td><td>Specifies the directories containing the topic and data files listed in the project file.</td></tr><tr><td><u>TITLE</u></td><td>Specifies the text displayed in the title bar of the Help window when the file is open.</td></tr><tr><td><u>WARNING</u></td><td>Specifies the level of error-message reporting the compiler is to display during the build.</td></tr></tbody></table> | Option | Description | <u>BMROOT</u> | Specifies the directory containing the bitmap files named in the bmc, bml, and bmr statements in topic files. This option is new for Windows 3.1. | <u>BUILD</u> | Specifies which topics to include in the build. | <u>CITATION</u> | Specifies a string that is appended to topics that are copied from Windows Help instead of the <u>COPYRIGHT</u> string. This option is new for Windows 3.1. | <u>COMPRESS</u> | Specifies the type of compression to use during the build. | <u>CONTENTS</u> | Specifies the context string of the Contents topic for a help file. This option is new for Windows 3.1. | <u>COPYRIGHT</u> | Adds a unique copyright message for the help file to the About dialog box. This option is new for Windows 3.1. | <u>ERRORLOG</u> | Puts compilation errors in a file during the build. This option is new for Windows 3.1. | <u>FORCEFONT</u> | Forces all authored fonts in the topic files to appear in a different font when displayed in the Help window. | <u>ICON</u> | Specifies the icon file to be displayed when the help file is minimized. This option is new for Windows 3.1. | <u>LANGUAGE</u> | Specifies a different sorting order for help files authored in a Scandinavian language. | <u>MAPFONTSIZE</u> | Maps a font size in the topic file to a different font size in the compiled help file. | <u>MULTIKEY</u> | Specifies an alternative keyword table to use for mapping topics. | <u>OLDKEYPHRASE</u> | Specifies whether the compiler should use the existing keyphrase table or create a new one during the build. This option is new for Windows 3.1. | <u>OPTCDROM</u> | Optimizes the help file for CD-ROM use. This option is new for Windows 3.1. | <u>REPORT</u> | Controls the display of messages during the build process. | <u>ROOT</u> | Specifies the directories containing the topic and data files listed in the project file. | <u>TITLE</u> | Specifies the text displayed in the title bar of the Help window when the file is open. | <u>WARNING</u> | Specifies the level of error-message reporting the compiler is to display during the build. |
| Option              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>BMROOT</u>       | Specifies the directory containing the bitmap files named in the bmc, bml, and bmr statements in topic files. This option is new for Windows 3.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>BUILD</u>        | Specifies which topics to include in the build.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>CITATION</u>     | Specifies a string that is appended to topics that are copied from Windows Help instead of the <u>COPYRIGHT</u> string. This option is new for Windows 3.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>COMPRESS</u>     | Specifies the type of compression to use during the build.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>CONTENTS</u>     | Specifies the context string of the Contents topic for a help file. This option is new for Windows 3.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>COPYRIGHT</u>    | Adds a unique copyright message for the help file to the About dialog box. This option is new for Windows 3.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>ERRORLOG</u>     | Puts compilation errors in a file during the build. This option is new for Windows 3.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>FORCEFONT</u>    | Forces all authored fonts in the topic files to appear in a different font when displayed in the Help window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>ICON</u>         | Specifies the icon file to be displayed when the help file is minimized. This option is new for Windows 3.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>LANGUAGE</u>     | Specifies a different sorting order for help files authored in a Scandinavian language.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>MAPFONTSIZE</u>  | Maps a font size in the topic file to a different font size in the compiled help file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>MULTIKEY</u>     | Specifies an alternative keyword table to use for mapping topics.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>OLDKEYPHRASE</u> | Specifies whether the compiler should use the existing keyphrase table or create a new one during the build. This option is new for Windows 3.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>OPTCDROM</u>     | Optimizes the help file for CD-ROM use. This option is new for Windows 3.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>REPORT</u>       | Controls the display of messages during the build process.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>ROOT</u>         | Specifies the directories containing the topic and data files listed in the project file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>TITLE</u>        | Specifies the text displayed in the title bar of the Help window when the file is open.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |
| <u>WARNING</u>      | Specifies the level of error-message reporting the compiler is to display during the build.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |        |             |               |                                                                                                                                                   |              |                                                 |                 |                                                                                                                                                             |                 |                                                            |                 |                                                                                                         |                  |                                                                                                                |                 |                                                                                         |                  |                                                                                                               |             |                                                                                                              |                 |                                                                                         |                    |                                                                                        |                 |                                                                   |                     |                                                                                                                                                  |                 |                                                                             |               |                                                            |             |                                                                                           |              |                                                                                         |                |                                                                                             |

### Comments

These options can appear in any order within the [OPTIONS] section. The [OPTIONS] section is not required.

## REPORT Option

REPORT = ON

The REPORT option displays messages on the screen during the build. These messages indicate when the Help compiler is performing the different phases of the build, including compiling the file, resolving jumps, and verifying browse sequences.

See Also

[OPTIONS], WARNING

## ROOT Option

ROOT = pathname[, pathname]..

The ROOT option specifies the directories where the Help compiler looks for files listed in the project file.

| Parameter | Description                                                                                                                                                                                                                                                                                                            |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pathname  | Specifies either a drive and full path or a relative path from the project directory. If the project file has a ROOT option, all relative paths in the project file refer to one of these paths. If the project file does not have a ROOT option, all paths are relative to the directory containing the project file. |

### Comments

If the project file does not have a BMROOT option, the compiler looks in the directories specified in the ROOT option to find bitmaps positioned by using the bmc, bml, and bmr statements. If none of these directories contains these bitmaps, the bitmap filenames must be listed in the [BITMAPS] section of the project file.

### Example

The following example specifies that the project root directory is C:\WINHELP\HELPPDIR and is found on drive C:

```
[ OPTIONS ]  
ROOT=C : \WINHELP\HELPPDIR
```

Given this root directory, if the [FILES] section contains the entry TOPICS\FILE.RTF, the full path for the topic file is C:\WINHELP\HELPPDIR\TOPICS\FILE.RTF.

### See Also

[BITMAPS], BMROOT, [OPTIONS]

## TITLE Option

TITLE = titlename

The TITLE option sets the title for the help file. Windows Help displays the title in its title bar whenever it displays the help file.

| Parameter | Description                                                                                           |
|-----------|-------------------------------------------------------------------------------------------------------|
| titlename | Specifies the title displayed in the Windows Help title bar. The title must not exceed 50 characters. |

### Comments

If no title is specified by using the TITLE option, Windows Help displays the title Windows Help in the title bar.

### Example

The following example sets the help-file title to ABC Help.

```
[OPTIONS]  
TITLE=ABC Help
```

See Also

[OPTIONS]



## WARNING Option

WARNING = level

The WARNING option specifies the amount of debugging information the Help compiler is to report.

| Parameter | Description                                                                                                                                                                                                                                                                        |       |         |   |                                     |   |                                          |   |                                 |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---|-------------------------------------|---|------------------------------------------|---|---------------------------------|
| level     | Specifies the warning level. This parameter may be one of the following values:                                                                                                                                                                                                    |       |         |   |                                     |   |                                          |   |                                 |
|           | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>1</td><td>Report only the most severe errors.</td></tr><tr><td>2</td><td>Report an intermediate number of errors.</td></tr><tr><td>3</td><td>Report all errors and warnings.</td></tr></tbody></table> | Value | Meaning | 1 | Report only the most severe errors. | 2 | Report an intermediate number of errors. | 3 | Report all errors and warnings. |
| Value     | Meaning                                                                                                                                                                                                                                                                            |       |         |   |                                     |   |                                          |   |                                 |
| 1         | Report only the most severe errors.                                                                                                                                                                                                                                                |       |         |   |                                     |   |                                          |   |                                 |
| 2         | Report an intermediate number of errors.                                                                                                                                                                                                                                           |       |         |   |                                     |   |                                          |   |                                 |
| 3         | Report all errors and warnings.                                                                                                                                                                                                                                                    |       |         |   |                                     |   |                                          |   |                                 |

### Example

The following example specifies an intermediate level of error reporting:

```
[ OPTIONS ]  
WARNING=2
```

See Also

[OPTIONS], REPORT

## [WINDOWS] Section

### [WINDOWS]

type = "caption", (x, y, width, height), sizing,  
(clientRGB), (nonscrollRGB), (fTop)

.  
.  
.

The [WINDOWS] section defines the size, location, and colors for the primary Help window and any secondary-window types used in a help file.

The secondary windows defined in this section are intended to be used with Windows applications that specify secondary windows when calling the WinHelp function.

| Parameter    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |   |                                                                                 |   |                                                                     |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---|---------------------------------------------------------------------------------|---|---------------------------------------------------------------------|
| type         | Specifies the type of window that uses the defined attributes. For the primary Help window, this parameter is main. For a secondary window, this parameter may be any unique name of up to 8 characters. Any jumps that display a topic in a secondary window give this type name as part of the jump.                                                                                                                                        |       |         |   |                                                                                 |   |                                                                     |
| caption      | Specifies the title for a secondary window. Windows Help places the title in the title bar of the window. To set the title for the primary Help window, use the <u>TITLE</u> option in the [OPTIONS] section.                                                                                                                                                                                                                                 |       |         |   |                                                                                 |   |                                                                     |
| x            | Specifies the x-coordinate, in help units, of the window's upper-left corner. Windows Help always assumes the screen is 1024 help units wide, regardless of resolution. For example, if the x-coordinate is 512, the left edge of the Help window is in the middle of the screen.                                                                                                                                                             |       |         |   |                                                                                 |   |                                                                     |
| y            | Specifies the y-coordinate, in help units, of the window's upper-left corner. Windows Help always assumes the screen is 1024 help units high, regardless of resolution. For example, if the x-coordinate is 512, the top edge of the Help window is in the middle of the screen.                                                                                                                                                              |       |         |   |                                                                                 |   |                                                                     |
| width        | Specifies the default width, in help units, for a secondary window.                                                                                                                                                                                                                                                                                                                                                                           |       |         |   |                                                                                 |   |                                                                     |
| height       | Specifies the default height, in help units, for a secondary window.                                                                                                                                                                                                                                                                                                                                                                          |       |         |   |                                                                                 |   |                                                                     |
| sizing       | Specifies the relative size of a secondary window when Windows Help first opens the window. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Set the window to the size specified by the x, y, width, and height parameters.</td></tr><tr><td>1</td><td>Maximize the window; ignore the x, y, width, and height parameters.</td></tr></tbody></table> | Value | Meaning | 0 | Set the window to the size specified by the x, y, width, and height parameters. | 1 | Maximize the window; ignore the x, y, width, and height parameters. |
| Value        | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |   |                                                                                 |   |                                                                     |
| 0            | Set the window to the size specified by the x, y, width, and height parameters.                                                                                                                                                                                                                                                                                                                                                               |       |         |   |                                                                                 |   |                                                                     |
| 1            | Maximize the window; ignore the x, y, width, and height parameters.                                                                                                                                                                                                                                                                                                                                                                           |       |         |   |                                                                                 |   |                                                                     |
| clientRGB    | Specifies the background color of the window. This parameter is an <u>RGB</u> color value consisting of three 8-bit hexadecimal numbers enclosed in parentheses and separated by commas. If this parameter is not given, Windows Help uses the default window color specified by Control Panel.                                                                                                                                               |       |         |   |                                                                                 |   |                                                                     |
| nonscrollRGB | Specifies the background color of the non-scrolling region (if any) in the Help window. This parameter is an <u>RGB</u> color value consisting of three 8-bit hexadecimal numbers enclosed in parentheses and separated by commas. If this parameter is not given, Windows Help uses the default window color specified by Control Panel.                                                                                                     |       |         |   |                                                                                 |   |                                                                     |
| fTop         | Specifies whether the secondary window is displayed on top of all other windows. When this parameter is 1, the window is displayed over all windows that do not also use this attribute. Otherwise, it should be zero. This parameter is optional.                                                                                                                                                                                            |       |         |   |                                                                                 |   |                                                                     |

### Example

The following example defines two windows, the main window and a secondary window named "picture". The main-window definition sets the background color of non-scrolling regions in the main Help window to (128, 0, 128) but leaves several other values empty (for which Windows Help will supply its own default values). The secondary-window definition sets the caption to "Samples" and sets the width and height of the window to about one-quarter of the width and height of the screen. The background colors for the window and non-scrolling region are (0, 255, 255) and (255, 0, 0), respectively. The sizing parameter for both the main and secondary windows is zero.

### [WINDOWS]

```
main=, ( , , , ), 0, ( , , ), (128, 0, 128)
```

picture = "Samples", (123,123,256,256), 0, (0,255,255), (255,0,0)

See Also

[Options], TITLE

## HPJ Statements

|                            |                                                           |
|----------------------------|-----------------------------------------------------------|
| <u>[ALIAS] Section</u>     | Assigns context strings to a topic alias                  |
| <u>[BAGGAGE] Section</u>   | Lists files to add to the Help file                       |
| <u>[BITMAPS] Section</u>   | Specifies the names of bitmap files                       |
| <u>BMROOT Option</u>       | Specifies the directory containing bitmaps                |
| <u>BUILD Option</u>        | Specifies which topics to build                           |
| <u>[BUILDTAGS] Section</u> | Specifies valid build tags                                |
| <u>CITATION Option</u>     | Inserts a citation string in the About dialog box         |
| <u>COMPRESS Option</u>     | Sets the level of compression for the help file           |
| <u>[CONFIG] Section</u>    | Specifies the Help file configuration                     |
| <u>CONTENTS Option</u>     | Specifies the context string of the contents topic        |
| <u>COPYRIGHT Option</u>    | Inserts a copyright string in the About dialog box        |
| <u>ERRORLOG Option</u>     | Specifies the file to receive error messages              |
| <u>[FILES] Section</u>     | Specifies the topic files                                 |
| <u>FORCEFONT Option</u>    | Sets the Help file font                                   |
| <u>ICON Option</u>         | Specifies the Windows Help icon                           |
| <u>LANGUAGE Option</u>     | Sets the sort-ordering for the keyword list               |
| <u>[MAP] Section</u>       | Associates context strings with context numbers           |
| <u>MAPEONTSIZE Option</u>  | Maps font sizes for the Help file                         |
| <u>MULTIKEY Option</u>     | Specifies the footnote for alternate keywords             |
| <u>OLDKEYPHRASE Option</u> | Specifies whether to use old phrase files                 |
| <u>OPTCDROM Option</u>     | Optimizes help file for display on CD-ROM                 |
| <u>[OPTIONS] Section</u>   | Contains options that control the Help compiler           |
| <u>REPORT Option</u>       | Displays build message during compilation                 |
| <u>ROOT Option</u>         | Specifies the directories containing topic and data files |
| <u>TITLE Option</u>        | Specifies the Help file title                             |
| <u>WARNING Option</u>      | Specifies the warning level for error messages            |
| <u>[WINDOWS] Section</u>   | Contains definitions for Help windows                     |

## About WinHelp macro

About()

The About macro displays Windows Help's About dialog box.

Parameters

This macro does not take any parameters.

Comments

Use of this macro in secondary windows is not recommended.

## AddAccelerator WinHelp macro

AddAccelerator(key, shift-state, "macro")

The AddAccelerator macro assigns a Help macro to an accelerator key (or key combination) so that the macro is carried out when the user presses the accelerator key(s).

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---|------|---|-------|---|------|---|------------|---|-----|---|-----------|---|----------|---|----------------|
| key         | Specifies the Windows virtual-key value. See the <a href="#">Virtual key codes</a> topic for a list of virtual-key codes that may be used for this parameter.                                                                                                                                                                                                                                                                                                                                                            |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| shift-state | Specifies the combination of ALT, SHIFT, and CTRL keys to be used with the accelerator. This parameter may be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>None</td></tr><tr><td>1</td><td>SHIFT</td></tr><tr><td>2</td><td>CTRL</td></tr><tr><td>3</td><td>SHIFT+CTRL</td></tr><tr><td>4</td><td>ALT</td></tr><tr><td>5</td><td>ALT+SHIFT</td></tr><tr><td>6</td><td>ALT+CTRL</td></tr><tr><td>7</td><td>SHIFT+ALT+CTRL</td></tr></tbody></table> | Value | Meaning | 0 | None | 1 | SHIFT | 2 | CTRL | 3 | SHIFT+CTRL | 4 | ALT | 5 | ALT+SHIFT | 6 | ALT+CTRL | 7 | SHIFT+ALT+CTRL |
| Value       | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 0           | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 1           | SHIFT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 2           | CTRL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 3           | SHIFT+CTRL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 4           | ALT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 5           | ALT+SHIFT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 6           | ALT+CTRL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 7           | SHIFT+ALT+CTRL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| macro       | Specifies the Help macro or macro string executed when the user presses the accelerator key(s). The macro must appear in quotation marks. Multiple macros in a string must be separated by semicolons.                                                                                                                                                                                                                                                                                                                   |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |

### Comments

The AddAccelerator macro can be abbreviated as AA.

### Example

The following macro executes the Windows Clock program when the user presses ALT+SHIFT+CONTROL+F4:

```
AddAccelerator(0x73, 7, "ExecProgram(`clock.exe', 1)")
```

### See Also

[RemoveAccelerator](#)

## Annotate WinHelp macro

Annotate()

The Annotate macro displays the Annotation dialog box from the Edit menu.

Parameters

This macro does not take any parameters.

Comments

Use of this macro in secondary windows is not recommended.

## AppendItem WinHelp macro

AppendItem("menu-id", "item-id", "item-name", "macro")

The AppendItem macro appends a menu item to the end of a menu created with the InsertMenu macro.

| Parameter | Description                                                                                                                                                                                                                     |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| menu-id   | Specifies the name used in the <u>InsertMenu</u> macro used to create the menu. This name must appear in quotation marks. The new item is appended to this menu.                                                                |
| item-id   | Specifies the name that Windows Help uses internally to identify the menu item. This name must appear in quotation marks. This name is used by the <u>DisableItem</u> or <u>DeleteItem</u> macros.                              |
| item-name | Specifies the name that Windows Help displays on the menu for the item. This name must appear in quotation marks. Within the quotation marks, place an ampersand (&) before the character used for the macro's accelerator key. |
| macro     | Specifies one or more macros that are to be executed when the user chooses the menu item. The macro must appear in quotation marks. Multiple macros in a string must be separated by semicolons (;).                            |

### Comments

Windows Help ignores this macro if it is executed in a secondary window.

If the keyboard accelerator conflicts with other menu access keys, Windows Help displays the error message "Unable to add item" and ignores the macro.

### Example

The following macro appends a menu item labeled "Tools" to a pop-up menu that has an identifier "IDM\_TLS". Choosing the menu item causes a jump to a topic with the context string "tpc1" in the TLS.HLP file:

```
AppendItem("IDM_BKS", "IDM_TLS", "&Tools", "JI(`tls.hlp', `tpc1')")
```

### See Also

DeleteItem, DisableItem, InsertMenu



## Back WinHelp macro

Back()

The Back macro displays the previous topic in the history list. The history list is a list of the last 40 topics the user has displayed since starting Windows Help.

Parameters

This macro does not take any parameters.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

If the Back macro is executed when the Back list is empty, Windows Help takes no action.

## BookmarkDefine WinHelp macro

BookmarkDefine()

The BookmarkDefine macro displays the Define dialog from the Bookmark menu.

Parameters

This macro does not take any parameters.

Comments

Use of this macro in secondary windows is not recommended.

If the BookmarkDefine macro is executed from a pop-up window, the bookmark is attached to the topic that invoked the pop-up window.

## BookmarkMore WinHelp macro

BookmarkMore()

The BookmarkMore macro displays the More dialog from the Bookmark menu. The More command appears on the Bookmark menu if the menu lists more than nine bookmarks.

### Parameters

This macro does not take any parameters.

### Comments

Use of the macro in secondary windows is not recommended.

## BrowseButtons WinHelp macro

BrowseButtons()

The BrowseButtons macro adds browse buttons to the button bar.

Parameters

This macro does not take any parameters.

Comments

Windows Help ignores this macro if it is executed from a secondary window.

If the BrowseButtons macro is used with one or more CreateButton macros in the [CONFIG] section of the project file, the order of the browse buttons on the Windows Help button bar is determined by the order of the BrowseButtons macro in relation to the other macros listed in the [CONFIG] section.

Example

The following macros in the project file cause the Clock button to appear immediately before the two browse buttons on the button bar:

```
[CONFIG]
CreateButton("&Clock", "ExecProgram(`clock', 0)")
BrowseButtons()
```

See Also

CreateButton

## ChangeButtonBinding WinHelp macro

`ChangeButtonBinding("button-id", "button-macro")`

The `ChangeButtonBinding` macro assigns a Help macro to a Help button.

| Parameter    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |    |             |              |          |            |        |          |      |             |         |              |                 |          |             |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|-------------|--------------|----------|------------|--------|----------|------|-------------|---------|--------------|-----------------|----------|-------------|
| button-id    | Specifies the identifier assigned to the button by the <code>CreateButton</code> macro or, for a standard Help button, one of the following predefined <u>button identifiers</u> : <table><thead><tr><th>ID</th><th>Description</th></tr></thead><tbody><tr><td>BTN_CONTENTS</td><td>Contents</td></tr><tr><td>BTN_SEARCH</td><td>Search</td></tr><tr><td>BTN_BACK</td><td>Back</td></tr><tr><td>BTN_HISTORY</td><td>History</td></tr><tr><td>BTN_PREVIOUS</td><td>Browse previous</td></tr><tr><td>BTN_NEXT</td><td>Browse next</td></tr></tbody></table> The button identifier must be enclosed in quotation marks. | ID | Description | BTN_CONTENTS | Contents | BTN_SEARCH | Search | BTN_BACK | Back | BTN_HISTORY | History | BTN_PREVIOUS | Browse previous | BTN_NEXT | Browse next |
| ID           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |    |             |              |          |            |        |          |      |             |         |              |                 |          |             |
| BTN_CONTENTS | Contents                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |             |              |          |            |        |          |      |             |         |              |                 |          |             |
| BTN_SEARCH   | Search                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |    |             |              |          |            |        |          |      |             |         |              |                 |          |             |
| BTN_BACK     | Back                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |    |             |              |          |            |        |          |      |             |         |              |                 |          |             |
| BTN_HISTORY  | History                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |    |             |              |          |            |        |          |      |             |         |              |                 |          |             |
| BTN_PREVIOUS | Browse previous                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |    |             |              |          |            |        |          |      |             |         |              |                 |          |             |
| BTN_NEXT     | Browse next                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |    |             |              |          |            |        |          |      |             |         |              |                 |          |             |
| button-macro | Specifies the Help macro executed when the user selects the button. The macro must be enclosed in quotation marks.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |    |             |              |          |            |        |          |      |             |         |              |                 |          |             |

### Comments

Windows Help ignores this macro if it is executed in a secondary window.

The `ChangeButtonBinding` macro can be abbreviated as `CBB`.

### Example

In the following macro, "conts" is the context string for the table of contents in the `DICT.HLP` file:

```
ChangeButtonBinding("btn_contents", "JumpId(`dict.hlp', `conts')")
```

## ChangeItemBinding WinHelp macro

```
ChangeItemBinding("item-id", "item-macro")
```

The ChangeItemBinding macro assigns a Help macro to an item previously added to a Windows Help menu using the AppendItem macro.

| Parameter  | Description                                                                                                                |
|------------|----------------------------------------------------------------------------------------------------------------------------|
| item-id    | Identifies the menu item appended by the <u>AppendItem</u> macro. The item identifier must be enclosed in quotation marks. |
| item-macro | Specifies the Help macro to execute when the user selects the item. The macro must be enclosed in quotation marks.         |

### Comments

Windows Help ignores this macro if it is executed in a secondary window.

The ChangeItemBinding macro can be abbreviated as CIB.

### Example

The following macro changes the menu item identified by "time\_item" so that it displays the Windows clock:

```
ChangeItemBinding("time_item", "ExecProgram(`clock', 0)")
```

## CheckItem WinHelp macro

CheckItem("item-id")

The CheckItem macro places a check-mark beside a menu item.

| Parameter | Description                                                                                 |
|-----------|---------------------------------------------------------------------------------------------|
| item-id   | Identifies the menu item to check. The item identifier must be enclosed in quotation marks. |

### Comments

The CheckItem macro can be abbreviated as CI.

### See Also

[UncheckItem](#)

## CloseWindow WinHelp macro

`CloseWindow("window-name")`

The `CloseWindow` macro closes either a secondary window or the main Help window.

| Parameter                | Description                                                                                                                                                                                                                                            |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>window-name</code> | Specifies the name of the window to close. The name "main" is reserved for the main Help window. For secondary windows, the window name is defined in the <u>[WINDOWS]</u> section of the project file. This name must be enclosed in quotation marks. |

### Example

The following macro closes the secondary window named "keys":

```
CloseWindow( "keys" )
```



## Contents WinHelp macro

### Contents()

The Contents macro displays the Contents topic in the current Help file. The Contents topic is defined by the CONTENTS option in the [OPTIONS] section of the project file. If the project file does not have a CONTENTS option, the Contents topic is the first topic of the first topic file specified in the project file.

## CopyDialog WinHelp macro

CopyDialog()

The CopyDialog macro displays the Copy dialog from the Edit menu.

### Comments

Use of this macro in secondary windows is not recommended.

## CopyTopic WinHelp macro

CopyTopic()

The CopyTopic macro copies all the text in the currently displayed topic to the Clipboard.

### Comments

Use of the macro in secondary windows is not recommended.

## CreateButton WinHelp macro

CreateButton("button-id", "name", "macro")

The CreateButton macro adds a new button to the button bar.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                            |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| button-id | Specifies the name that WinHelp uses internally to identify the button. This name must appear in quotation marks. Use this name in the <a href="#">DisableButton</a> or <a href="#">DestroyButton</a> macro if you want to remove or disable the button or in the <a href="#">ChangeButtonBinding</a> if you want to change the Help macro that the button executes in certain topics. |
| name      | Specifies the text that appears on the button. To make a letter in this text the mnemonic for the button, place an ampersand (&) before that letter. The button name is case-sensitive and can have up to 29 characters in it -- any additional characters are ignored.                                                                                                                |
| macro     | Specifies the Help macro or macro string executed when the user clicks on the button. Multiple macros in a macro string must be separated by semicolons.                                                                                                                                                                                                                               |

### Comments

Windows Help allows a maximum of 16 custom buttons. It allows a total of 22 buttons, including the standard Browse buttons, on the button bar.

If the [BrowseButtons](#) macro is used with one or more CreateButton macros in the project file, the buttons appear in the same order on the button bar as the macros appear in the project file.

Windows Help ignores this macro if it is executed in a secondary window.

The CreateButton macro can be abbreviated as CB.

### Example

The following macro creates a new button labeled "Ideas" that jumps to the topic with the context string "dir" in the IDEAS.HLP file when clicked:

```
CreateButton("btn_ideas", "&Ideas", "JumpId(`ideas.hlp', `dir')")
```

### See Also

[DisableButton](#), [DestroyButton](#), [ChangeButtonBinding](#), [JumpId](#)

## DeleteItem WinHelp macro

DeleteItem("item-id")

The DeleteItem macro removes a menu item that was added by using the AppendItem macro.

| Parameter | Description                                                                                                                 |
|-----------|-----------------------------------------------------------------------------------------------------------------------------|
| item-id   | Specifies the item identifier used in the <u>AppendItem</u> macro. The item identifier must be enclosed in quotation marks. |

### Comments

Windows Help ignores this macro if it is executed in a secondary window.

### Example

The following macro removes the menu item "Tools" appended in the example for the AppendItem macro:

```
DeleteItem( "IDM_TOOLS" )
```

### See Also

AppendItem

## DeleteMark WinHelp macro

DeleteMark("marker-text")

The DeleteMark macro removes a text marker added with the SaveMark macro.

| Parameter   | Description                                                                                                                   |
|-------------|-------------------------------------------------------------------------------------------------------------------------------|
| marker-text | Specifies the text marker previously added by the <u>SaveMark</u> macro. The marker text must be enclosed in quotation marks. |

### Comments

If the marker does not exist when the DeleteMark macro is executed, Windows Help displays a "Topic not found" error message.

### Example

The following macro removes the marker "Managing Memory" from a Help file:

```
DeleteMark("Managing Memory")
```

### See Also

SaveMark

## DestroyButton WinHelp macro

DestroyButton("button-id")

The DestroyButton macro removes a button added with the CreateButton macro.

| Parameter | Description                                                                                                                         |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------|
| button-id | Identifies a button previously created by the <u>CreateButton</u> macro. The button identifier must be enclosed in quotation marks. |

### Comments

The button identifier cannot be an identifier for one of the standard Help buttons. For a list of those identifiers, see the ChangeButtonBinding macro.

Windows Help ignores this macro if it is executed in a secondary window.

### See Also

CreateButton, ChangeButtonBinding

## DisableButton WinHelp macro

DisableButton("button-id")

The DisableButton macro grays out a button added with the CreateButton macro. This button cannot be used in the topic until an EnableButton macro is executed.

| Parameter | Description                                                                                                                                  |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------|
| button-id | Specifies the identifier assigned to the button by the <u>CreateButton</u> macro. The button identifier must be enclosed in quotation marks. |

### Comments

Windows Help ignores this macro if it is executed in a secondary window.

The DisableButton macro can be abbreviated as DB.

### See Also

CreateButton, EnableButton



## DisableItem WinHelp macro

DisableItem("item-id")

The DisableItem macro grays out a menu item added with the AppendItem macro. The menu item cannot be used in the topic until an EnableItem macro is executed.

| Parameter | Description                                                                                                                           |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------|
| item-id   | Identifies a menu item previously appended with the <u>AppendItem</u> macro. The item identifier must be enclosed in quotation marks. |

### Comments

Windows Help ignores this macro if it is executed in a secondary window.

The DisableItem macro can be abbreviated as DI.

### See Also

AppendItem

## EnableButton WinHelp macro

EnableButton("button-id")

The EnableButton macro re-enables a button disabled with the DisableButton macro.

| Parameter | Description                                                                                                                                  |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------|
| button-id | Specifies the identifier assigned to the button by the <u>CreateButton</u> macro. The button identifier must be enclosed in quotation marks. |

### Comments

Windows Help ignores this macro if it is executed in a secondary window.

The EnableButton macro can be abbreviated as EB.

### See Also

CreateButton, DisableButton

## EnableItem WinHelp macro

EnableItem("item-id")

The EnableItem macro re-enables a menu item disabled with the DisableItem macro.

| Parameter | Description                                                                                                                                 |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| item-id   | Specifies the identifier assigned to the menu item by the <u>AppendItem</u> macro. The item identifier must be enclosed in quotation marks. |

### Comments

Windows Help ignores this macro if it is executed in a secondary window.

The EnableItem macro can be abbreviated as EI.

### See Also

AppendItem, DisableItem

## ExecProgram WinHelp macro

ExecProgram("command-line", display-state)

The ExecProgram macro executes a Windows application.

| Parameter     | Description                                                                                                                                                                                                                                                                                                      |       |         |   |        |   |           |   |           |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---|--------|---|-----------|---|-----------|
| command-line  | Specifies the command line for the application to be executed. The command line must be enclosed in quotation marks. Windows Help searches for this application in the current directory, followed by the Windows directory, the user's path, and the directory of the currently viewed Help file.               |       |         |   |        |   |           |   |           |
| display-state | Specifies a value indicating how the application is shown when executed. It may be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Normal</td></tr><tr><td>1</td><td>Minimized</td></tr><tr><td>2</td><td>Maximized</td></tr></tbody></table> | Value | Meaning | 0 | Normal | 1 | Minimized | 2 | Maximized |
| Value         | Meaning                                                                                                                                                                                                                                                                                                          |       |         |   |        |   |           |   |           |
| 0             | Normal                                                                                                                                                                                                                                                                                                           |       |         |   |        |   |           |   |           |
| 1             | Minimized                                                                                                                                                                                                                                                                                                        |       |         |   |        |   |           |   |           |
| 2             | Maximized                                                                                                                                                                                                                                                                                                        |       |         |   |        |   |           |   |           |

### Comments

The ExecProgram macro can be abbreviated as EP.

The backslash character should not be used to escape double quotation-mark characters in macros. Instead, you can enclose the command line in single quotation marks and omit the backslash for the double quotation marks, as shown in the following:

```
`command "string as parameter"'
```

Note that the first single quotation mark must be an open quote and the last single quotation mark must be a close quote.

### Example

The following example executes the Clock application. The application is minimized when it starts:

```
ExecProgram(`clock.exe', 1)
```

## Exit WinHelp macro

Exit()

The Exit macro exits the Windows Help application. It has the same effect as selecting Exit from the File menu.

Parameters

This macro does not take any parameters.

## FileOpen WinHelp macro

FileOpen()

The FileOpen macro displays the Open dialog box from the File menu.

Parameters

This macro does not take any parameters.

Comments

Use of the macro in secondary windows is not recommended.

## FocusWindow WinHelp macro

FocusWindow("window-name")

The FocusWindow macro changes the focus to the specified window, either the main Help window or a secondary window.

| Parameter   | Description                                                                                                                                                                                                                                                        |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| window-name | Specifies the name of the window to receive the focus. The name "main" is reserved for the main Help window. For secondary windows, the window name is defined in the <u>[WINDOWS]</u> section of the project file. This name must be enclosed in quotation marks. |

### Comments

This macro is ignored if the specified window does not exist.

### Example

The following macro changes the focus to the secondary window "keys":

```
FocusWindow( "keys" )
```

## GoToMark WinHelp macro

GoToMark("marker-text")

The GoToMark macro jumps to a marker set with the SaveMark macro.

| Parameter   | Description                                                                    |
|-------------|--------------------------------------------------------------------------------|
| marker-text | Specifies a text marker previously defined by using the <u>SaveMark</u> macro. |

### Example

The following macros jumps to the marker "Managing Memory".

```
GoToMark("Managing Memory")
```

See Also  
SaveMark



## HelpOn WinHelp macro

### HelpOn()

The HelpOn macro displays the Help file for the Windows Help application. The macro carries out the same action as choosing the How to Use Help command on the Help menu.

### Parameters

This macro does not take any parameters.

## HelpOnTop WinHelp macro

HelpOnTop()

The HelpOnTop macro toggles the on-top state of Windows Help. It is equivalent to checking or unchecking the Always On Top command in the Help menu.

Parameters

This macro does not take any parameters.

Comments

Windows Help does not provide a macro to check the current state of the Always On Top command. It is up to the user to determine whether the macro should be used to change the state of the command.

## History WinHelp macro

### History()

The History macro displays the history list, which shows the last 40 topics the user has viewed since opening a Help file in Windows Help. It has the same effect as choosing the History button.

### Parameters

This macro does not take any parameters.

### Comments

Windows Help ignores this macro if it is executed in a secondary window.

## IfThen WinHelp macro

`IfThen(IsMark("marker-text"), "macro")`

The IfThen macro executes a Help macro if a given marker exists. It uses the IsMark macro to make the test.

| Parameter   | Description                                                                                                                                    |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| marker-text | Specifies a text marker previously created by using the <u>SaveMark</u> macro. The marker must be enclosed in quotation marks.                 |
| macro       | Specifies a Help macro or macro string to be executed if the marker exists. Multiple macros in a macro string must be separated by semicolons. |

### Example

The following macro jumps to the topic with context string "man\_mem" if a marker named "Managing Memory" has been set by the SaveMark macro:

```
IfThen(IsMark("Managing Memory"), "JI(`trb.hlp', `man_mem')")
```

See Also

IsMark, SaveMark

## IfThenElse WinHelp macro

`IfThenElse(IsMark("marker-text"), "macro1", "macro2")`

The IfThenElse macro executes one of two Help macros depending on whether or not a marker exists. It uses the IsMark macro to make the test.

| Parameter   | Description                                                                                                                                                 |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| marker-text | Specifies a text marker previously created by using the <u>IsMark</u> macro. The marker must be enclosed in quotation marks.                                |
| macro1      | Specifies a Help macro or macro string to be executed if the marker exists. Multiple macros in either macro string must be separated by semicolons.         |
| macro2      | Specifies a Help macro or macro string to be executed if the marker does not exist. Multiple macros in either macro string must be separated by semicolons. |

### Example

The following macro jumps to the topic with context string "mem" if a marker named "Memory" has been set by the SaveMark macro. If the marker does not exist, it jumps to the next topic in the browse sequence.

```
IfThenElse(IsMark("Memory"), "JI(`trb.hlp', `mem')", "Next()")
```

### See Also

IfThen, IsMark, SaveMark

## InsertItem WinHelp macro

InsertItem("menu-id", "item-id", "item-name", "macro", position)

The InsertItem macro inserts a menu item at a given position on an existing menu. The menu can be either one you create with the InsertMenu macro or one of the standard Windows Help menus.

| Parameter    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |      |      |          |      |          |      |              |               |            |      |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|----------|------|----------|------|--------------|---------------|------------|------|
| menu-id      | Identifies either a standard Windows Help menu or a menu previously created by using the <u>InsertMenu</u> macro. For a standard menu, this parameter can be one of the following: <table><thead><tr><th>Name</th><th>Menu</th></tr></thead><tbody><tr><td>MNU_FILE</td><td>File</td></tr><tr><td>MNU_EDIT</td><td>Edit</td></tr><tr><td>MNU_BOOKMARK</td><td>Bookmark menu</td></tr><tr><td>MNU_HELPON</td><td>Help</td></tr></tbody></table> For other menus, this parameter must be the name used with the <u>InsertMenu</u> macro. In all cases, the menu identifier must be enclosed in quotation marks. The new item is inserted into this menu. | Name | Menu | MNU_FILE | File | MNU_EDIT | Edit | MNU_BOOKMARK | Bookmark menu | MNU_HELPON | Help |
| Name         | Menu                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |      |      |          |      |          |      |              |               |            |      |
| MNU_FILE     | File                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |      |      |          |      |          |      |              |               |            |      |
| MNU_EDIT     | Edit                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |      |      |          |      |          |      |              |               |            |      |
| MNU_BOOKMARK | Bookmark menu                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |      |      |          |      |          |      |              |               |            |      |
| MNU_HELPON   | Help                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |      |      |          |      |          |      |              |               |            |      |
| item-id      | Specifies the name that Windows Help uses internally to identify the menu item. The item identifier must be enclosed in quotation marks.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |      |      |          |      |          |      |              |               |            |      |
| item-name    | Specifies the name Windows Help displays in the menu for the item. This name is case-sensitive and must be enclosed in quotation marks. An ampersand (&) before a character in the name identifies it as the item's keyboard access key.                                                                                                                                                                                                                                                                                                                                                                                                               |      |      |          |      |          |      |              |               |            |      |
| macro        | Specifies a Help macro or macro string to be executed when the user chooses the menu item. The macro must be enclosed in quotation marks. Multiple macros in a string must be separated by semicolons (;).                                                                                                                                                                                                                                                                                                                                                                                                                                             |      |      |          |      |          |      |              |               |            |      |
| position     | Specifies the position of the menu item in the menu. It must be an integer value. Position 0 is the first or topmost position in the menu.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |      |      |          |      |          |      |              |               |            |      |

### Comments

The item-id parameter can be used in a subsequent DisableItem or DeleteItem macro to remove or disable the item or to change the operations that the item performs in certain topics.

Windows Help ignores this macro if it is executed in a secondary window.

The specified keyboard access keys must be unique. If a key conflicts with other menu access keys, Windows Help displays the error message "Unable to add item" and ignores the macro.

### Example

The following macro inserts a menu item labeled "Tools" as the third item on a menu that has an identifier "MNU\_BKS". Selecting the menu item causes a jump to a topic with the context string "tls1" in the TLS.HLP file:

```
InsertItem("mnu_bks", "m_tls", "&Tools", "JI(`tls.hlp', `tls1')", 3)
```

### See Also

InsertMenu

## InsertMenu WinHelp macro

InsertMenu("menu-id", "menu-name", menu-position)

The InsertMenu inserts a new menu in the Windows Help menu bar.

| Parameter     | Description                                                                                                                                                                                                               |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| menu-id       | Specifies the name that Windows Help uses internally to identify the menu. The menu identifier must be enclosed in quotation marks. This identifier can be used in the <u>AppendItem</u> macro to add macros to the menu. |
| menu-name     | Specifies the name that Windows Help displays on the menu bar. This name must be enclosed in quotation marks. An ampersand (&) before a character in the name identifies it as the menu's keyboard access key.            |
| menu-position | Specifies the position on the menu bar of the new menu name. This parameter must be an integer number. Positions are numbered from left to right, with position 0 the left-most menu.                                     |

### Comments

Windows Help ignores this macro if it is executed in a secondary window.

### Example

The following macro adds a menu named "Utilities" to the Windows Help application. The label "Utilities" appears as the fourth item on the Windows Help menu bar. The user presses U with the ALT key to open the menu.

```
InsertMenu("IDM_UTIL", "&Utilities", 3)
```

### See Also

AppendItem, InsertItem

## IsMark WinHelp macro

IsMark("marker-text")

The IsMark macro tests whether or not a marker set by the SaveMark macro exists. It is used as a parameter to the conditional macros IfThen and IfThenElse. The IsMark macro returns nonzero if the mark exists or zero if it does not.

| Parameter   | Description                                                               |
|-------------|---------------------------------------------------------------------------|
| marker-text | Specifies a text marker previous created using the <u>SaveMark</u> macro. |

### Comments

The Not macro can be used to reverse the results of the IsMark macro.

### Example

The following macro jumps to the topic with the context string "man\_mem" if a marker named "Managing Memory" has been set by the SaveMark macro:

```
IfThen(IsMark("Managing Memory"), "JI(`trb.hlp', `man_mem')")
```

### See Also

IfThen, IfThenElse, Not



## JumpContents WinHelp macro

JumpContents("filename")

The JumpContents macro jumps to the Contents topic of a specified file in the Help file. The Contents topic is indicated by the CONTENTS option entry in the [OPTIONS] section of project file. If the CONTENTS option is not specified, Windows Help jumps to the first topic in the Help file.

| Parameter | Description                                                                                                                                                                                                   |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename  | Specifies the name of the destination file for the jump. The filename must be enclosed in quotation marks. If Windows Help cannot find this file, it displays an error message and does not perform the jump. |

### Comments

Windows Help ignores this macro if it is executed in a secondary window.

### Example

The following macro jumps to the Contents topic of the PROGMAN.HLP file:

```
JumpContents ( "PROGMAN.HLP" )
```

See Also

JumpContext

## JumpContext WinHelp macro

JumpContext("filename", context-number)

| Parameter      | Description                                                                                                                                                                                                                                                                                           |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename       | Specifies the name of the destination file for the jump. The filename must be enclosed in quotation marks. If Windows Help cannot find this file, it displays an error message and does not perform the jump.                                                                                         |
| context-number | Specifies the context number of the topic in the destination file. The context number must be defined in the [MAP] section of the project file. If the context number is not valid, Windows Help jumps to the Contents topic or to the first topic in the file instead and displays an error message. |

### Comments

The JumpContext macro can be abbreviated as JC.

### Example

The following macro jumps to the topic mapped to the context number 801 in the PROGMAN.HLP file:

```
JumpContext("PROGMAN.HLP", 801)
```

### See Also

[JumpContents](#)

## JumpHelpOn WinHelp macro

### JumpHelpOn()

The JumpHelpOn macro jumps to the Contents topic of the How to Use Help file. The How To Use Help file is either the default WINHELP.HLP file shipped with Windows 3.1 or the Help file designated by the SetHelpOnFile macro in the [CONFIG] section of the project file.

### Parameters

This macro does not take any parameters.

### Comments

If Windows Help cannot find the specified Help file, it displays an error message and does not perform the jump.

### Example

The following macro jumps to the Contents topic of the designated How to Use Help file:

```
JumpHelpOn( )
```

## JumpId WinHelp macro

`JumpId("filename", "context-string")`

The `JumpId` macro jumps to the topic with the specified context string in the Help file.

| Parameter      | Description                                                                                                                                                                                                               |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename       | Specifies the name of the Help file containing the context string. The filename must be enclosed in quotation marks. If Windows Help does not find this file, it displays an error message and does not perform the jump. |
| context-string | Context string of the topic in the destination file. The context string must be enclosed in quotation marks. If the context string does not exist, Windows Help jumps to the Contents topic for that file instead.        |

### Comments

The `JumpId` macro may be abbreviated as `JJ`.

### Example

The following macro jumps to a topic with "second\_topic" as its context string in the `SECOND.HLP` file:

```
JJ("second.hlp", "second_topic")
```

## JumpKeyword WinHelp macro

`JumpKeyword("filename", "keyword")`

The `JumpKeyword` macro loads the indicated Help file, searches through the K keyword table, and displays the first topic containing the index keyword specified in the macro.

| Parameter | Description                                                                                                                                                                                                                                                                                                                 |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename  | Specifies the name of the Help file containing the desired keyword table. The filename must be enclosed in quotation marks. If this file does not exist, Windows Help displays an error message and does not perform the jump.                                                                                              |
| keyword   | Specifies the keyword that the macro searches for. The keyword must be enclosed in quotation marks. If Windows Help finds more than one match, it displays the first matched topic. If it does not find any matches, it displays a "Not a keyword" message and displays the Contents topic of the destination file instead. |

### Comments

The `JumpKeyword` macro can be abbreviated as `JK`.

### Example

The following macro displays the first topic that has "hands" as an index keyword in the `CLOCK.HLP` file:

```
JumpKeyword("clock.hlp", "hands")
```

## Next WinHelp macro

Next()

The Next macro displays the next topic in the browse sequence for the Help file.

Parameters

This macro does not take any parameters.

Comments

If the currently displayed topic is the last topic of a browse sequence, this macro does nothing.

Windows Help ignores this macro if it is executed in a secondary window.

## Not WinHelp macro

`Not(IsMark("marker-text"))`

The Not macro reverses the result (nonzero or zero) returned by the IsMark macro. It is used along with the IsMark macro as a parameter to the conditional macros IfThen and IfThenElse.

| Parameter   | Description                                                                                                                         |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------|
| marker-text | Specifies a text marker previously created by using the <u>SaveMark</u> macro. The marker text must be enclosed in quotation marks. |

### Example

The following macro jumps to the topic with the context string "mem1" if a marker named "Memory" has not been set by the SaveMark macro:

```
IfThen(Not(IsMark("Memory")), "JI(`trb.hlp', `mem1')")
```

### See Also

IfThen, IfThenElse, IsMark

## PopupContext WinHelp macro

PopupContext("filename", context-number)

The PopupContext macro displays in a pop-up window the topic identified by a specific context number.

| Parameter      | Description                                                                                                                                                                                                                                                    |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename       | Specifies the name of the file that contains the topic to be displayed. The filename must be enclosed in quotation marks. If Windows Help cannot find this file, it displays an error message.                                                                 |
| context number | Specifies the context number of the topic to be displayed. The context number must be specified in the [MAP] section of the project file. If the context number is not valid, Windows Help displays the Contents topic or the first topic in the file instead. |

### Comments

The PopupContext macro can be abbreviated as PC.

### Example

The following macro displays in a pop-up window the topic mapped to the context number 801 in the PROGMAN.HLP file:

```
PopupContext("progman.hlp", 801)
```

### See Also

PopupId



## PopupId WinHelp macro

`PopupId("filename", "context-string")`

The `PopupId` macro displays a topic from a specified file in a pop-up window.

| Parameter      | Description                                                                                                                                                                                 |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename       | Specifies the name of the file containing the pop-up window topic. The filename must be enclosed in quotation marks. If this file does not exist, Windows Help displays a warning.          |
| context-string | Specifies the context string of the topic in the destination file. If the requested context string does not exist, Windows Help displays the Contents topic or the first topic in the file. |

### Comments

The `PopupId` macro can be abbreviated as `PI`.

### Example

The following macro displays a pop-up window with context string "second\_topic" from the `SECOND.HLP` file:

```
PopupId("second.hlp", "second_topic")
```

### See Also

[PopupContext](#)

## PositionWindow WinHelp macro

PositionWindow(x, y, width, height, state, "name")

The PositionWindow macro sets the size and position of a window.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                |       |         |   |             |   |           |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---|-------------|---|-----------|
| x         | Specifies the x-coordinate, in help units, of the upper-left corner of the window. Windows Help always assumes the screen (regardless of resolution) is 1024 help units wide. For example, if the x-coordinate is 512, the left edge of the Help window is in the middle of the screen.                                                    |       |         |   |             |   |           |
| y         | Specifies the y-coordinate, in help units, of the upper-left corner of the window. Windows Help always assumes the screen (regardless of resolution) is 1024 help units high. For example, if the y-coordinate is 512, the top edge of the Help window is in the middle of the screen.                                                     |       |         |   |             |   |           |
| width     | Specifies the default width, in help units, of the window.                                                                                                                                                                                                                                                                                 |       |         |   |             |   |           |
| height    | Specifies the default height, in help units, of the window.                                                                                                                                                                                                                                                                                |       |         |   |             |   |           |
| state     | Specifies how the window is sized. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Normal size</td></tr><tr><td>1</td><td>Maximized</td></tr></tbody></table> If the parameter is 1, Windows Help ignores the x, y, width, and height parameters. | Value | Meaning | 0 | Normal size | 1 | Maximized |
| Value     | Meaning                                                                                                                                                                                                                                                                                                                                    |       |         |   |             |   |           |
| 0         | Normal size                                                                                                                                                                                                                                                                                                                                |       |         |   |             |   |           |
| 1         | Maximized                                                                                                                                                                                                                                                                                                                                  |       |         |   |             |   |           |
| name      | Specifies the name of the window to position. The name "main" is reserved for the main Help window. For secondary windows, the window name must be defined in the [WINDOWS] section of the project file. This name must be enclosed in quotation marks.                                                                                    |       |         |   |             |   |           |

### Comments

If the window to be positioned does not exist, Windows Help ignores the macro.

The PositionWindow macro can be abbreviated as PW.

### Example

The following macro positions the secondary window "Samples" in the upper-left corner (100, 100) with a width and height of 500 (in help units):

```
PositionWindow(100, 100, 500, 500, 0, "Samples")
```

## Prev WinHelp macro

Prev()

The Prev macro displays the previous topic in the browse sequence for the Help file. If the currently displayed topic is the first topic of a browse sequence, this macro does nothing.

Parameters

This macro does not take any parameters.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

## Print WinHelp macro

Print()

The Print macro sends the currently displayed topic to the printer. It should be used only to print topics in windows other than the main Help window (for example, topics in a secondary window).

Parameters

This macro does not take any parameters.

See Also

[PrinterSetup](#)

## PrinterSetup WinHelp macro

PrinterSetup()

The PrinterSetup macro displays the Printer Setup dialog box from the File menu.

Parameters

This macro does not take any parameters.

Comments

Use of the macro in secondary windows is not recommended.

See Also

Print

## RegisterRoutine WinHelp macro

RegisterRoutine("DLL-name", "function-name", "format-spec")

The RegisterRoutine macro registers a function within a dynamic-link library (DLL). Registered functions can be used in macro footnotes in topic files or in the [CONFIG] section of the project file, the same as standard Help macros.

| Parameter     | Description                                                                                                                                                                                  |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DLL-name      | Specifies the filename of the DLL. The filename must be enclosed in quotation marks. If Windows Help cannot find the library, it displays an error message.                                  |
| function-name | Specifies the name of the function to execute in the designated DLL.                                                                                                                         |
| format-spec   | Specifies a string indicating the formats of parameters passed to the function. The format string must be enclosed in quotation marks. Characters in the string represent C parameter types: |

| Character | Description                    |
|-----------|--------------------------------|
| u         | unsigned short ( <u>WORD</u> ) |
| U         | unsigned long ( <u>DWORD</u> ) |
| i         | short int                      |
| I         | int                            |
| s         | near char * ( <u>PSTR</u> )    |
| S         | far char * ( <u>LPSTR</u> )    |
| v         | void                           |

If the function is used as a Help macro, Windows Help makes sure the macro parameters match the parameter types given in this macro.

### Comments

The RegisterRoutine macro can be abbreviated as RR.

### Example

The following call registers a routine named PlayAudio in a DLL, MMLIB.DLL. PlayAudio takes arguments of the far char \*, int, and unsigned long types:

```
RegisterRoutine("MMLIB", "PlayAudio", "SIU")
```

## RemoveAccelerator WinHelp macro

RemoveAccelerator(key, shift-state)

The RemoveAccelerator macro removes the assignment of a Help macro to an accelerator key (or key combination). These assignments are made by using the AddAccelerator macro.

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---|------|---|-------|---|------|---|------------|---|-----|---|-----------|---|----------|---|----------------|
| key         | Specifies the Windows virtual-key value. See the <u>Virtual key codes</u> topic for a list of virtual-key codes that may be used for this parameter.                                                                                                                                                                                                                                                                                                                                                                         |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| shift-state | Specifies the combination of ALT, SHIFT, and CTRL keys that were used with the accelerator. This parameter may be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>None</td></tr><tr><td>1</td><td>SHIFT</td></tr><tr><td>2</td><td>CTRL</td></tr><tr><td>3</td><td>SHIFT+CTRL</td></tr><tr><td>4</td><td>ALT</td></tr><tr><td>5</td><td>ALT+SHIFT</td></tr><tr><td>6</td><td>ALT+CTRL</td></tr><tr><td>7</td><td>SHIFT+ALT+CTRL</td></tr></tbody></table> | Value | Meaning | 0 | None | 1 | SHIFT | 2 | CTRL | 3 | SHIFT+CTRL | 4 | ALT | 5 | ALT+SHIFT | 6 | ALT+CTRL | 7 | SHIFT+ALT+CTRL |
| Value       | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 0           | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 1           | SHIFT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 2           | CTRL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 3           | SHIFT+CTRL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 4           | ALT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 5           | ALT+SHIFT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 6           | ALT+CTRL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |
| 7           | SHIFT+ALT+CTRL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |   |      |   |       |   |      |   |            |   |     |   |           |   |          |   |                |

### Comments

The RemoveAccelerator macro can be abbreviated as RA. No error occurs when this macro is used with an accelerator for which a macro was not defined.

### Example

The following macro disassociates a macro from the ALT+SHIFT+CONTROL+F4 key combination:

```
RemoveAccelerator(0x73, 7)
```

### See Also

AddAccelerator

## SaveMark WinHelp macro

SaveMark("marker-text")

The SaveMark macro saves the location of the currently displayed topic and file and associates a text marker with that location. The GotoMark macro can then be used to jump to this location.

| Parameter   | Description                                                                                                                                                                                                                             |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| marker-text | Specifies the text marker to be used to identify the topic location. This text must be enclosed in quotation marks, and it must be unique. If the same text is used for more than one marker, the most recently entered marker is used. |

### Comments

A text marker can be used with the GotoMark, DeleteMark, IfThen, and IfThenElse macros.

If the user exits Windows Help, all text markers are deleted.

### Example

The following macro saves the marker "Managing Memory" in the current topic:

```
SaveMark("Managing Memory")
```

### See Also

DeleteMark, GotoMark, IfThen, IfThenElse, IsMark, Not



## Search WinHelp macro

Search()

The Search macro displays the dialog for the Search button, which allows users to search for topics using keywords defined by the K footnote character.

Parameters

This macro does not take any parameters.

Comments

Windows Help ignores this macro if it is executed in a secondary window.

## SetContents WinHelp macro

SetContents("filename", context-number)

The SetContents macro designates a specific topic as the Contents topic in the specified Help file.

| Parameter      | Description                                                                                                                                                                                                                |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename       | Specifies the name of the Help file that contains the Contents topic. The filename must be enclosed in quotation marks. If Windows Help cannot find this file, it displays an error message and does not perform the jump. |
| context number | Specifies the context number of the topic in the specified file. The context number must be defined in the [MAP] section of the project file. If the context number is not valid, Windows Help displays an error message.  |

### Example

The following example sets the topic mapped to the context number 801 in the PROGMAN.HLP file as the Contents topic. After executing this macro, clicking the Contents button will cause a jump to the topic specified by the context-number parameter:

```
SetContents("PROGMAN.HLP", 801)
```

## SetHelpOnFile WinHelp macro

SetHelpOnFile("filename")

| Parameter | Description                                                                                                                                                                        |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename  | Specifies the name of the replacement How to Use Help file. The filename must be enclosed in quotation marks. If Windows Help cannot find this file, it displays an error message. |

### Comments

If this macro appears in a topic in the Help file, the replacement file is set after execution of the macro. If this macro appears in the [CONFIG] section of the project file, the replacement file is set when the help file is opened.

### Example

The following macro sets the Using Help file to MYHELP.HLP:

```
SetHelpOnFile("myhelp.hlp")
```

## UncheckItem WinHelp macro

UncheckItem("item-id")

The UncheckItem macro removes the check mark from a menu item.

| Parameter | Description                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------|
| item-id   | Identifies the menu item to uncheck. The item identifier must be enclosed in quotation marks. |

### Comments

The UncheckItem macro can be abbreviated UI.

### See Also

[CheckItem](#)

## Help Macros

|                                                          |                                                |
|----------------------------------------------------------|------------------------------------------------|
| <a href="#"><u>About WinHelp macro</u></a>               | Displays the About dialog box                  |
| <a href="#"><u>AddAccelerator WinHelp macro</u></a>      | Assigns a macro to an accelerator key          |
| <a href="#"><u>Annotate WinHelp macro</u></a>            | Displays Annotation dialog box                 |
| <a href="#"><u>AppendItem WinHelp macro</u></a>          | Appends a menu item                            |
| <a href="#"><u>Back WinHelp macro</u></a>                | Displays previous topic in the history list    |
| <a href="#"><u>BookmarkDefine WinHelp macro</u></a>      | Displays the Define dialog box                 |
| <a href="#"><u>BookmarkMore WinHelp macro</u></a>        | Displays the More dialog box                   |
| <a href="#"><u>BrowseButtons WinHelp macro</u></a>       | Adds browse buttons                            |
| <a href="#"><u>ChangeButtonBinding WinHelp macro</u></a> | Assigns a macro to a button                    |
| <a href="#"><u>ChangeItemBinding WinHelp macro</u></a>   | Assigns a macro to a menu item                 |
| <a href="#"><u>CheckItem WinHelp macro</u></a>           | Checks a menu item                             |
| <a href="#"><u>CloseWindow WinHelp macro</u></a>         | Closes a window                                |
| <a href="#"><u>Contents WinHelp macro</u></a>            | Displays the Contents topic                    |
| <a href="#"><u>CopyDialog WinHelp macro</u></a>          | Displays the Copy dialog box                   |
| <a href="#"><u>CopyTopic WinHelp macro</u></a>           | Copies current topic to the clipboard          |
| <a href="#"><u>CreateButton WinHelp macro</u></a>        | Adds a new button to the button bar            |
| <a href="#"><u>DeleteItem WinHelp macro</u></a>          | Removes a menu item                            |
| <a href="#"><u>DeleteMark WinHelp macro</u></a>          | Deletes a text marker                          |
| <a href="#"><u>DestroyButton WinHelp macro</u></a>       | Removes a button from the button bar           |
| <a href="#"><u>DisableButton WinHelp macro</u></a>       | Disables a button                              |
| <a href="#"><u>DisableItem WinHelp macro</u></a>         | Disables a menu item                           |
| <a href="#"><u>EnableButton WinHelp macro</u></a>        | Enables a button                               |
| <a href="#"><u>EnableItem WinHelp macro</u></a>          | Enables a menu item                            |
| <a href="#"><u>ExecProgram WinHelp macro</u></a>         | Executes a program                             |
| <a href="#"><u>Exit WinHelp macro</u></a>                | Exits WinHelp                                  |
| <a href="#"><u>FileOpen WinHelp macro</u></a>            | Displays the Open dialog box                   |
| <a href="#"><u>FocusWindow WinHelp macro</u></a>         | Changes the focus window                       |
| <a href="#"><u>GoToMark WinHelp macro</u></a>            | Jumps to a marker                              |
| <a href="#"><u>HelpOn WinHelp macro</u></a>              | Displays the Help on Using topic               |
| <a href="#"><u>HelpOnTop WinHelp macro</u></a>           | Toggles on-top state of help                   |
| <a href="#"><u>History WinHelp macro</u></a>             | Displays the history list                      |
| <a href="#"><u>IfThen WinHelp macro</u></a>              | Executes macro if marker exists                |
| <a href="#"><u>IfThenElse WinHelp macro</u></a>          | Executes one of two macros if marker exists    |
| <a href="#"><u>InsertItem WinHelp macro</u></a>          | Inserts a menu item                            |
| <a href="#"><u>InsertMenu WinHelp macro</u></a>          | Inserts a new menu                             |
| <a href="#"><u>IsMark WinHelp macro</u></a>              | Tests if a marker is set                       |
| <a href="#"><u>JumpContents WinHelp macro</u></a>        | Jumps to the Contents topic                    |
| <a href="#"><u>JumpContext WinHelp macro</u></a>         | Jumps to the specified context                 |
| <a href="#"><u>JumpHelpOn WinHelp macro</u></a>          | Jumps to Using Help file                       |
| <a href="#"><u>JumpId WinHelp macro</u></a>              | Jumps to the specified topic                   |
| <a href="#"><u>JumpKeyword WinHelp macro</u></a>         | Jumps to the topic containing the keyword      |
| <a href="#"><u>Next WinHelp macro</u></a>                | Displays the next topic in the browse sequence |
| <a href="#"><u>Not WinHelp macro</u></a>                 | Reverses the IsMark macro                      |
| <a href="#"><u>PopupContext WinHelp macro</u></a>        | Displays a topic in a popup window             |
| <a href="#"><u>PopupId WinHelp macro</u></a>             | Displays topic in a popup window               |
| <a href="#"><u>PositionWindow WinHelp macro</u></a>      | Sets the size and position of a window         |
| <a href="#"><u>Prev WinHelp macro</u></a>                | Displays previous topic in browse sequence     |
| <a href="#"><u>Print WinHelp macro</u></a>               | Prints the current topic                       |
| <a href="#"><u>PrinterSetup WinHelp macro</u></a>        | Displays the Printer Setup dialog box          |
| <a href="#"><u>RegisterRoutine WinHelp macro</u></a>     | Registers a DLL function                       |
| <a href="#"><u>RemoveAccelerator WinHelp macro</u></a>   | Assigns a macro to an accelerator key          |
| <a href="#"><u>SaveMark WinHelp macro</u></a>            | Saves a marker                                 |
| <a href="#"><u>Search WinHelp macro</u></a>              | Displays the Search dialog box                 |
| <a href="#"><u>SetContents WinHelp macro</u></a>         | Sets the Contents topic                        |
| <a href="#"><u>SetHelpOnFile WinHelp macro</u></a>       | Sets the Using Help help file                  |
| <a href="#"><u>UncheckItem WinHelp macro</u></a>         | Unchecks a menu item                           |

## RTF Tokens

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <u>\ansi</u>      | Specifies the ANSI character set                  |
| <u>\b</u>         | Starts bold text                                  |
| <u>\bin</u>       | Specifies binary picture data                     |
| <u>\bmc</u>       | Displays a bitmap or metafile in text             |
| <u>\bml</u>       | Displays a bitmap or metafile at the left margin  |
| <u>\bmr</u>       | Displays a bitmap or metafile at the right margin |
| <u>\box</u>       | Draws a box                                       |
| <u>\brdrb</u>     | Draws a bottom border                             |
| <u>\brdrbar</u>   | Draws a vertical bar                              |
| <u>\brdrdb</u>    | Sets double-lined borders                         |
| <u>\brdrdot</u>   | Sets dotted border                                |
| <u>\brdrl</u>     | Draws a left border                               |
| <u>\brdrr</u>     | Draws a right border                              |
| <u>\brdrs</u>     | Sets standard borders                             |
| <u>\brdrt</u>     | Draws a top border                                |
| <u>\brdrth</u>    | Sets thick borders                                |
| <u>\cell</u>      | Marks end of table cell                           |
| <u>\cellx</u>     | Sets the position of a cell's right edge          |
| <u>\cf</u>        | Sets the foreground color                         |
| <u>\colortbl</u>  | Creates the color table                           |
| <u>\deff</u>      | Sets default font                                 |
| <u>\emc</u>       | Allows DLL to paint window in text                |
| <u>\eml</u>       | Allows DLL to paint window at left margin         |
| <u>\emr</u>       | Allows DLL to paint window at right margin        |
| <u>\f</u>         | Sets the font                                     |
| <u>\fi</u>        | Sets the first-line indent                        |
| <u>\fldrslt</u>   | Result of a field                                 |
| <u>\fonttbl</u>   | Creates the font table                            |
| <u>\footnote</u>  | Defines topic-specific information                |
| <u>\fs</u>        | Sets the font size                                |
| <u>\v</u>         | Inserts a character by value                      |
| <u>\i</u>         | Starts italic text                                |
| <u>\intbl</u>     | Marks paragraph as in table                       |
| <u>\keep</u>      | Makes text non-wrapping                           |
| <u>\keepn</u>     | Creates a non-scrolling region                    |
| <u>\li</u>        | Sets the left indent                              |
| <u>\line</u>      | Breaks the current line                           |
| <u>\mac</u>       | Sets the Apple MacIntosh character set            |
| <u>\page</u>      | Ends current topic                                |
| <u>\par</u>       | Marks the end of a paragraph                      |
| <u>\pard</u>      | Restores default paragraph properties             |
| <u>\pc</u>        | Sets the PC character set                         |
| <u>\pich</u>      | Specifies the picture height                      |
| <u>\pichgoal</u>  | Specifies the desired picture height              |
| <u>\picscalex</u> | Specifies the horizontal scaling value            |
| <u>\picscaley</u> | Specifies the vertical scaling value              |
| <u>\pict</u>      | Creates a picture                                 |
| <u>\picw</u>      | Specifies the picture width                       |
| <u>\picwgoal</u>  | Specifies the desired picture width               |
| <u>\plain</u>     | Restores default character properties             |
| <u>\qc</u>        | Centers text                                      |

|                       |                                            |
|-----------------------|--------------------------------------------|
| <u>\ql</u>            | Aligns text left                           |
| <u>\qr</u>            | Aligns text right                          |
| <u>\ri</u>            | Sets the right indent                      |
| <u>\row</u>           | Marks end of a table row                   |
| <u>\rtf</u>           | Specifies the RTF version                  |
| <u>\sa</u>            | Sets the spacing after a paragraph         |
| <u>\sb</u>            | Sets space before                          |
| <u>\scaps</u>         | Starts small capitals                      |
| <u>\sect</u>          | Marks the end of a section and paragraph   |
| <u>\sl</u>            | Sets the spacing between lines             |
| <u>\strike</u>        | Creates a hotspot                          |
| <u>\tab</u>           | Inserts a tab character                    |
| <u>\tqc</u>           | Tabs and centers text                      |
| <u>\tqr</u>           | Tabs and aligns text right                 |
| <u>\trgaph</u>        | Sets space between text columns in a table |
| <u>\trleft</u>        | Sets left margin for the first cell        |
| <u>\trowd</u>         | Sets table defaults                        |
| <u>\trqc</u>          | Sets relative column widths                |
| <u>\trql</u>          | Left-aligns table row                      |
| <u>\tx</u>            | Sets a tab stop                            |
| <u>\ul</u>            | Creates a link to a pop-up topic           |
| <u>\uldh</u>          | Creates a hot spot                         |
| <u>\v</u>             | Creates a link to a topic                  |
| <u>\whitmap</u>       | Specifies a Windows bitmap                 |
| <u>\wbmbitspixel</u>  | Specifies the number of bits per pixel     |
| <u>\wbmplanes</u>     | Specifies the number of planes             |
| <u>\wbmwidthbytes</u> | Specifies the bitmap width in bytes        |
| <u>\wmetafile</u>     | Specifies a Windows metafile               |

## `\ansi` RTF statement

`\ansi`

The `\ansi` statement sets the American National Standards Institute (ANSI) character set. The Windows character set is essentially equivalent to the ANSI character set.

See Also

`\mac`, `\pc`



## **\b** RTF statement

**\b**

The **\b** statement starts bold text. The statement applies to all subsequent text up to the next **\plain** or **\b0** statement.

### Comments

No **\plain** or **\b0** statement is required if the **\b** statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

The **\b0** statement was first supported in the Microsoft Help Compiler version 3.1.

### Example

The following example sets "Note" to bold:

```
{\b Note} Setting the Auto option frees novice users from  
determining their system configurations.
```

### See Also

**\i**, **\plain**, **\scaps**

## `\bin` RTF statement

### `\bin`

The `\bin` statement indicates the start of binary picture data. The Help compiler interprets subsequent bytes in the file as binary data. This statement is used in conjunction with the `\pict` statement.

| Parameter      | Description                                                           |
|----------------|-----------------------------------------------------------------------|
| <code>n</code> | Specifies the number of bytes of binary data following the statement. |

### Comments

A single space character must separate the `\bin` statement from subsequent bytes. The Microsoft Help Compiler assumes that all subsequent bytes, including linefeed and carriage return characters, are binary data. These bytes can have any value in the range 0 through 255. For this reason, the `\bin` statement is typically used in program-generated files only.

If the `\bin` statement is not given with a `\pict` statement, the default picture data format is hexadecimal.

### See Also

`\pict`

## bmc RTF statement

\{bmc filename\}

The bmc statement displays a specified bitmap or metafile in the current line of text. The statement positions the bitmap or metafile as if it were the next character in the line, aligning it on the base line and applying the current paragraph properties.

| Parameter | Description                                                                                                                                              |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename  | Specifies the name of a file containing either a Windows bitmap, a placeable Windows metafile, a multiresolution bitmap, or a segmented-graphics bitmap. |

### Comments

Since the bmc statement is not a standard RTF statement, the Microsoft Help Compiler relies on the opening and closing braces, including the backslashes (\), to distinguish the statement from regular text.

If a file containing a metafile is specified, the file must contain a placeable Windows metafile; the Microsoft Help Compiler will not accept standard Windows metafiles. Furthermore, Windows Help sets the MM\_ANISOTROPIC mode prior to displaying the metafile, so the placeable Windows metafile must either set the window origin and extents or set some other mapping mode.

### See Also

[bmr](#), [bml](#), [\wbitmap](#)

## bml RTF statement

\{bml filename\}

The bml statement displays a specified bitmap or metafile at the left margin of the Help window. The first line of subsequent text aligns with the upper-right corner of the image and subsequent lines wrap along the right edge of the image.

| Parameter | Description                                                                                                                                              |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename  | Specifies the name of a file containing either a Windows bitmap, a placeable Windows metafile, a multiresolution bitmap, or a segmented-graphics bitmap. |

### Comments

Since the bml statement is not a standard RTF statement, the Microsoft Help Compiler relies on the opening and closing braces, including the backslashes (\), to distinguish the statement from regular text.

If a file containing a metafile is specified, the file must contain a placeable Windows metafile; the Microsoft Help Compiler will not accept standard Windows metafiles. Furthermore, Windows Help sets the MM\_ANISOTROPIC mode prior to displaying the metafile, so the placeable Windows metafile must either set the window origin and extents or set some other mapping mode.

### See Also

[bmc](#), [bmr](#), [\wbitmap](#)

## bmr RTF statement

\{bmr filename\}

The bmr statement displays a specified bitmap or metafile at the right margin of the Help window. The first line of subsequent text aligns with the upper-left corner of the image and subsequent lines wrap along the left edge of the image.

| Parameter | Description                                                                                                                                              |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename  | Specifies the name of a file containing either a Windows bitmap, a placeable Windows metafile, a multiresolution bitmap, or a segmented-graphics bitmap. |

### Comments

Since the bmr statement is not a standard RTF statement, the Microsoft Help Compiler relies on the opening and closing braces, including the backslashes (\), to distinguish the statement from regular text.

If a file containing a metafile is specified, the file must contain a placeable Windows metafile; the Help compiler will not accept standard Windows metafiles. Furthermore, Windows Help sets the MM\_ANISOTROPIC mode prior to displaying the metafile, so the placeable Windows metafile must either set the window origin and extents or set some other mapping mode.

### See Also

[bmc](#), [bml](#), [\wbitmap](#)

## \box RTF statement

### \box

The \box statement draws a box around the current paragraph or picture. The statement applies to all subsequent paragraphs or pictures up to the next \pard statement.

### Comments

For paragraphs, Windows Help uses the height of the paragraph, excluding space before or after the paragraph, as the height of the box. For pictures (as defined by \pict statements), Windows Help uses the specified height of the picture as the height of the box. For both paragraphs and pictures, the width of the box is equal to the space between the left and right indents.

Windows Help draws the box using the current border style.

### Example

The following example draws a box around the paragraph:

```
\par \box  
{\b Note} Setting the Auto option frees novice users from  
determining their system configurations.  
\par \pard
```

### See Also

\brdrb, \brdl, \brdr, \brdt, \pard

## `\brdrb` RTF statement

### `\brdrb`

The `\brdrb` statement draws a border below the current paragraph or picture. The statement applies to all subsequent paragraphs or pictures up to the next `\pard` statement.

### Comments

Windows Help draws the border using the current border style.

### See Also

`\box`, `\brdrbar`, `\brdl`, `\brdr`, `\brdt`, `\pard`

## `\brdrbar` RTF statement

### `\brdrbar`

The `\brdrbar` statement draws a vertical bar to the left of the current paragraph or picture. The statement applies to all subsequent paragraphs or pictures up to the next `\pard` statement.

### Comments

Windows Help draws the border using the current border style.

In a print-based document, the `\brdrbar` statement draws the bar on the right side of paragraphs on odd-numbered pages, but on the left side of paragraphs on even-numbered pages.

### See Also

`\box`, `\brdl`, `\brdrb`, `\brdr`, `\brdt`, `\pard`



`\brdrdb` RTF statement

`\brdrdb`

The `\brdrdb` statement selects a double line for drawing borders. The selection applies to all subsequent paragraphs or pictures up to the next `\pard` statement.

See Also

`\brdrdot`, `\brdrs`, `\brdrth`, `\pard`

`\brdrdot` RTF statement

`\brdrdot`

The Help compiler ignores this statement.

See Also

`\brdrs`, `\brdrth`, `\brdrdb`, `\pard`

## `\brdl` RTF statement

`\brdl`

The `\brdl` statement draws a border to the left of the current paragraph or picture. The statement applies to all subsequent paragraphs or pictures up to the next `\pard` statement.

Comments

Windows Help draws the border using the current border style.

See Also

`\box`, `\brdrb`, `\brdrbar`, `\brdr`, `\brdt`, `\pard`

## `\brdr` RTF statement

### `\brdr`

The `\brdr` statement draws a border to the right of the current paragraph or picture. The statement applies to all subsequent paragraphs or pictures up to the next `\pard` statement.

### Comments

Windows Help draws the border using the current border style.

### See Also

`\box`, `\brdrb`, `\brdrbar`, `\brdl`, `\brdrf`, `\pard`

## `\brdrs` RTF statement

`\brdrs`

The `\brdrs` statement selects a standard-width line for drawing borders. The selection applies to all subsequent paragraphs or pictures up to the next `\pard` statement.

See Also

`\brdrdb`, `\brdrdot`, `\brdrth`, `\pard`

## `\brdr` RTF statement

### `\brdr`

The `\brdr` statement draws a border above the current paragraph or picture. The statement applies to all subsequent paragraphs or pictures up to the next `\pard` statement.

### Comments

Windows Help draws the border using the current border style.

### See Also

`\box`, `\brdrb`, `\brdrbar`, `\brdrl`, `\brdrr`, `\pard`

`\bdrth` RTF statement

`\bdrth`

The `\bdrth` statement selects a thick line for drawing borders. The selection applies to all subsequent paragraphs or pictures up to the next `\pard` statement.

See Also

`\bdrdb`, `\bdrdot`, `\bdrs`, `\pard`

## \cell RTF statement (3.1)

### \cell

The \cell statement marks the end of a cell in a table. A cell consists of all paragraphs from a preceding \intbl or \cell statement to the ending \cell statement. Windows Help formats and displays these paragraphs using the left and right margins of the cell and any current paragraph properties.

### Comments

This statement was first supported in the Microsoft Help Compiler version 3.1.

### Example

The following example creates a two-column table. The second column contains three separate paragraphs, each having different paragraph properties:

```
\cellx2880\cellx5760  
\intbl  
Alignment\cell  
\ql  
Left-aligned  
\par  
\qc  
Centered  
\par  
\qr  
Right-aligned\cell  
\row \pard
```

### See Also

\cellx, \intbl, \row, \trgaph, \trleft, \trowd



## \cellx RTF statement (3.1)

### \cellxn

The \cellx statement sets the absolute position of the right edge of a table cell. One \cellx statement must be given for each cell in the table. The first \cellx statement applies to the left-most cell, the last to the right-most cell. For each \cellx statement, the specified position applies to the corresponding cell in each subsequent row of the table up to the next \trowd statement.

| Parameter | Description                                                                                                                                                         |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| n         | Specifies the position of the cell's right edge, in twips. The position is relative to the left edge of the Help window. It is not affected by the current indents. |

### Comments

A table consists of a grid of cells in columns and rows. Each cell has an explicitly defined right edge; the position of a cell's left edge is the same as the position of the right edge of the adjacent cell. For the left-most cell in a row, the left edge position is equal to the Help window's left margin position. Each cell has a left and right margin between which Windows Help aligns and wraps text. By default, the margin positions are equal to the left and right edges. The \trgaph and \trleft statements can be used to set different margins for all cells in a row.

This statement was first supported in the Microsoft Help Compiler version 3.1.

### Example

The following example creates a three-column table having two rows. The positions of the right edges of the three cells are 2, 4, and 6 inches, respectively:

```
\cellx2880\cellx5760\cellx8640
\intbl
Row 1 Cell 1\cell
Row 1 Cell 2\cell
Row 1 Cell 3\cell
\row
\intbl
Row 2 Cell 1\cell
Row 2 Cell 2\cell
Row 2 Cell 3\cell
\row \pard
```

### See Also

\cell, \intbl, \row, \trgaph, \trleft, \trowd

## \cf RTF statement

### \cfn

The \cf statement sets the foreground color. The new color applies to all subsequent text up to the next \plain or \cf statement.

| Parameter | Description                                                                                                                                                                                                                                                                   |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| n         | Specifies the color number to set as foreground. The number must be an integer number in the range 1 to the maximum number of colors specified in the color table for the Help file. If an invalid color number is specified, Windows Help uses the default foreground color. |

### Comments

No \plain or \cf statement is required if the \cf statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to the enclosed text only.

If the \cf statement is not given, the default foreground color is the text color set by Control Panel.

### Example

The following example displays green text:

```
{\colortbl;\red0\green255\blue0;}
{\cf1 This text is green.}
```

### See Also

\colortbl

## `\colortbl` RTF statement

```
{\colortbl
\redr\greeng\blueb;
.
.
.
}
```

The `\colortbl` statement creates a color table for the Help file. The color table consists of one or more color definitions. Each color definition consists of one `\red`, `\green`, and `\blue` statement specifying the amount of primary color to use to generate the final color. Each color definition must end with a semicolon (;).

| Parameter | Description                                                                                      |
|-----------|--------------------------------------------------------------------------------------------------|
| r         | Specifies the intensity of red in the color. It must be an integer in the range 0 through 255.   |
| g         | Specifies the intensity of green in the color. It must be an integer in the range 0 through 255. |
| b         | Specifies the intensity of blue in the color. It must be an integer in the range 0 through 255.  |

### Comments

Color definitions are implicitly numbered starting at zero. A color definition's implicit number can be used in the `\cf` statement to set the foreground color.

The default colors are the window-text and window-background colors set by Control Panel. To override the default colors, both a `\colortbl` statement and a `\cf` statement must be given.

### Example

The following example creates a color table containing two color definitions. The first color definition is empty (only the semicolon is given), so color number 0 always represents the default color. The second definition specifies green; color number 1 can be used to display green text:

```
{\colortbl;\red0\green255\blue0;}
```

### See Also

`\cf`

## \deff RTF statement

### \defn

The \deff statement sets the default font number. Windows Help uses the number to set the default font whenever a \plain statement is given or an invalid font number is given in a \f statement.

| Parameter | Description                                                                                                                                                                  |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| n         | Specifies the number of the font to be used as the default font. This parameter must be a valid font number as specified by the <u>\fonttbl</u> statement for the Help file. |

### Comments

If the \deff statement is not given, the default font number is zero.

### See Also

\f, \fonttbl, \plain

## emc RTF statement

\{emc module, class, data [, dx, dy]\}

The emc statement allows an external dynamic-link library to paint a window that is embedded in a Help topic. This statement displays the window in the current line of text. The statement positions the window as if it were the next character in the line, aligning it on the base line and applying the current paragraph properties.

| Parameter | Description                                                                               |
|-----------|-------------------------------------------------------------------------------------------|
| module    | Specifies the name of the dynamic-link library that paints the embedded window.           |
| class     | Specifies the name of the registered window class for the embedded window.                |
| data      | Specifies a string that is passed to the embedded window in its <u>WM_CREATE</u> message. |
| dx        | Specifies the suggested width of the embedded window. This parameter is optional.         |
| dy        | Specifies the suggested height of the embedded window. This parameter is optional.        |

### Comments

Since the emc statement is not a standard RTF statement, the Microsoft Help Compiler relies on the opening and closing braces, including the backslashes (\), to distinguish the statement from regular text.

### See Also

bmr, bml, bmc, eml, emr, \wbitmap

## eml RTF statement

\{eml module, class, data [, dx, dy]\}

The eml statement allows an external dynamic-link library to paint a window that is embedded at the left margin in a Help topic. The first line of subsequent text aligns with the upper-right corner of the window and subsequent lines wrap along the right edge of the window.

| Parameter | Description                                                                               |
|-----------|-------------------------------------------------------------------------------------------|
| module    | Specifies the name of the dynamic-link library that paints the embedded window.           |
| class     | Specifies the name of the registered window class for the embedded window.                |
| data      | Specifies a string that is passed to the embedded window in its <u>WM_CREATE</u> message. |
| dx        | Specifies the suggested width of the embedded window. This parameter is optional.         |
| dy        | Specifies the suggested height of the embedded window. This parameter is optional.        |

### Comments

Since the eml statement is not a standard RTF statement, the Microsoft Help Compiler relies on the opening and closing braces, including the backslashes (\), to distinguish the statement from regular text.

### See Also

bmr, bml, bmc, emc, emr, \wbimap

## emr RTF statement

\{emr module, class, data [, dx, dy]\}

The emr statement allows an external dynamic-link library to paint a window that is embedded at the right margin in a Help topic. The first line of subsequent text aligns with the upper-left corner of the window and subsequent lines wrap along the left edge of the window.

| Parameter | Description                                                                               |
|-----------|-------------------------------------------------------------------------------------------|
| module    | Specifies the name of the dynamic-link library that paints the embedded window.           |
| class     | Specifies the name of the registered window class for the embedded window.                |
| data      | Specifies a string that is passed to the embedded window in its <u>WM_CREATE</u> message. |
| dx        | Specifies the suggested width of the embedded window. This parameter is optional.         |
| dy        | Specifies the suggested height of the embedded window. This parameter is optional.        |

### Comments

Since the emr statement is not a standard RTF statement, the Microsoft Help Compiler relies on the opening and closing braces, including the backslashes (\), to distinguish the statement from regular text.

### See Also

bmr, bml, bmc, emc, eml, \wbitmap

## `\f` RTF statement

`\fn`

The `\f` statement sets the font. The new font applies to all subsequent text up to the next [`\plain`](#) or `\f` statement.

| Parameter      | Description                                                                                                                    |
|----------------|--------------------------------------------------------------------------------------------------------------------------------|
| <code>n</code> | Specifies the font number. This parameter must be one of the integer font numbers defined in the font table for the Help file. |

### Comments

The `\f` statement does not set the point size of the font; use the [`\fs`](#) statement instead.

No `\plain` or `\f` statement is required if the `\f` statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

If the `\f` statement is not given, the default font is defined by the `\deff` statement (or is zero if no [`\deff`](#) statement is given).

### Example

The following example uses the Arial font to display text:

```
{\fonttbl {\f0\fswiss Arial;}}
{\f0
This text illustrates the Arial font.}
\par
```

### See Also

[`\deff`](#), [`\fonttbl`](#), [`\fs`](#), [`\plain`](#)



## `\fi` RTF statement

`\fin`

The `\fi` statement sets the first-line indent for the paragraph. The new indent applies to the first line of each subsequent paragraph up to the next `\pard` statement or `\fi` statement. The first-line indent is always relative to the current left indent.

| Parameter      | Description                                                                                 |
|----------------|---------------------------------------------------------------------------------------------|
| <code>n</code> | Specifies the indent, in twips. This parameter can be either a positive or negative number. |

### Comments

If the `\fi` statement is not given, the first-line indent is zero by default.

### Example

The following example uses the first-line indent and a tab stop to make a numbered list:

```
\tx360\li360\fi-360
1
\tab
Insert the disk in drive A.
\par
2
\tab
Type a:setup and press the ENTER key.
\par
3
\tab
Follow the instructions on the screen.
\par \pard
```

### See Also

`\li`, `\pard`

## `\fldrslt` RTF statement

### `\fldrslt`

The `\fldrslt` statement specifies the most recently calculated result of a field. The Microsoft Help Compiler interprets the result as text and formats it using the current character and paragraph properties.

### Comments

The Help compiler ignores all field statements except the `\fldrslt` statement. Any text associated with other field statements is ignored.

## \fonttbl RTF statement

```
{\fonttbl
  {\fn\family font-name;}
  .
  .
}
```

The \fonttbl statement creates a font table for the Help file. The font table consists of one or more font definitions. Each definition consists of a font number, a font family, and a font name.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |      |                                    |        |                                                                               |        |                                                                    |         |                                                                                |         |                                     |        |                                                                   |       |                                                                 |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|------|------------------------------------|--------|-------------------------------------------------------------------------------|--------|--------------------------------------------------------------------|---------|--------------------------------------------------------------------------------|---------|-------------------------------------|--------|-------------------------------------------------------------------|-------|-----------------------------------------------------------------|
| n         | Specifies the font number. This parameter must be an integer. This number can be used in subsequent \f statements to set the current font to the specified font. In the font table, font numbers should start at zero and increase by one for each new font definition.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |      |                                    |        |                                                                               |        |                                                                    |         |                                                                                |         |                                     |        |                                                                   |       |                                                                 |
| family    | Specifies the font family. This parameter must be one of the following: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>fnil</td><td>Unknown or default fonts (default)</td></tr><tr><td>froman</td><td>Roman, proportionally spaced serif fonts (for example, MS Serif and Palatino)</td></tr><tr><td>fswiss</td><td>Swiss, proportionally spaced sans serif fonts (for example, Swiss)</td></tr><tr><td>fmodern</td><td>Fixed-pitch serif and sans serif fonts (for example, Courier, Elite, and Pica)</td></tr><tr><td>fscript</td><td>Script fonts (for example, Cursive)</td></tr><tr><td>fdecor</td><td>Decorative fonts (for example, Old English and ITC Zapf Chancery)</td></tr><tr><td>ftech</td><td>Technical, symbol, and mathematical fonts (for example, Symbol)</td></tr></tbody></table> | Value | Meaning | fnil | Unknown or default fonts (default) | froman | Roman, proportionally spaced serif fonts (for example, MS Serif and Palatino) | fswiss | Swiss, proportionally spaced sans serif fonts (for example, Swiss) | fmodern | Fixed-pitch serif and sans serif fonts (for example, Courier, Elite, and Pica) | fscript | Script fonts (for example, Cursive) | fdecor | Decorative fonts (for example, Old English and ITC Zapf Chancery) | ftech | Technical, symbol, and mathematical fonts (for example, Symbol) |
| Value     | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |      |                                    |        |                                                                               |        |                                                                    |         |                                                                                |         |                                     |        |                                                                   |       |                                                                 |
| fnil      | Unknown or default fonts (default)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |      |                                    |        |                                                                               |        |                                                                    |         |                                                                                |         |                                     |        |                                                                   |       |                                                                 |
| froman    | Roman, proportionally spaced serif fonts (for example, MS Serif and Palatino)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |      |                                    |        |                                                                               |        |                                                                    |         |                                                                                |         |                                     |        |                                                                   |       |                                                                 |
| fswiss    | Swiss, proportionally spaced sans serif fonts (for example, Swiss)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |      |                                    |        |                                                                               |        |                                                                    |         |                                                                                |         |                                     |        |                                                                   |       |                                                                 |
| fmodern   | Fixed-pitch serif and sans serif fonts (for example, Courier, Elite, and Pica)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |      |                                    |        |                                                                               |        |                                                                    |         |                                                                                |         |                                     |        |                                                                   |       |                                                                 |
| fscript   | Script fonts (for example, Cursive)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |      |                                    |        |                                                                               |        |                                                                    |         |                                                                                |         |                                     |        |                                                                   |       |                                                                 |
| fdecor    | Decorative fonts (for example, Old English and ITC Zapf Chancery)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |      |                                    |        |                                                                               |        |                                                                    |         |                                                                                |         |                                     |        |                                                                   |       |                                                                 |
| ftech     | Technical, symbol, and mathematical fonts (for example, Symbol)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |      |                                    |        |                                                                               |        |                                                                    |         |                                                                                |         |                                     |        |                                                                   |       |                                                                 |
| font-name | Specifies the name of the font. This parameter should specify an available Windows font.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |      |                                    |        |                                                                               |        |                                                                    |         |                                                                                |         |                                     |        |                                                                   |       |                                                                 |

### Comments

If a font with the specified name is not available, Windows Help chooses a font from the specified family. If no font from the given family exists, Windows Help chooses a font having the same character set as specified for the Help file.

The \deff statement sets the default font number for the Help file. The default font is set whenever the \pard statement is given.

### See Also

[\deff](#), [\f](#), [\fs](#), [\pard](#)

## \footnote RTF statement

{n}{\footnote {n} text}

The \footnote statement defines topic-specific information, such as the topic's build tags, context string, title, browse number, keywords, and execution macros. Every topic must have a context string, at least, to give the user access to the topic through links.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |                                                                                                                                                                                                                                                                                                                                 |    |                                                                                                                                                                                                    |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |   |                                                                                                                                |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|--------------------------------------------------------------------------------------------------------------------------------|
| n         | Specifies the footnote character. It can be one of the following: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>*</td><td>Specifies a build tag. The Microsoft Help Compiler uses build tags to determine whether it should include the topic in the Help file. The text parameter can be any combination of characters but must not contain spaces. Uppercase and lowercase characters are treated as equivalent characters (case-insensitive). If a topic has build-tag statements, they must be the first statements in the topic. The Microsoft Help Compiler checks a topic for build tags if the project file specifies a build expression using the <u>BUILD</u> option.</td></tr><tr><td>#</td><td>Specifies a context string. The text parameter can be any combination of letters and digits but must not contain spaces. Uppercase and lowercase characters are treated as equivalent characters (case-insensitive). The context string can be used with the <u>\v</u> statement in other topics to create links to this topic.</td></tr><tr><td>\$</td><td>Specifies a topic title. Windows Help uses the topic title to identify the topic in the Search and History dialog boxes. The text parameter can be any combination of characters including spaces.</td></tr><tr><td>+</td><td>Specifies the browse-sequence identifier. Windows Help adds topics having an identifier to the browse sequence and allows users to view the topics by using the browse buttons. The text parameter can be a combination of letters and digits. Windows Help determines the order of topics in the browse sequence by sorting the identifier alphabetically. If two topics have the same identifier, Windows Help assumes that the topic that was compiled first is to be displayed first. Windows Help uses the browse sequence identifier only if the browse buttons have been enabled by using the <u>BrowseButtons</u> macro.</td></tr><tr><td>K</td><td>Specifies a keyword. Windows Help displays all keywords in the Help file in the Search dialog box and allows a user to choose a topic to view by choosing a keyword. The text parameter can be any combination of characters including spaces. If the first character is the letter K, it must be preceded with an extra space or a semicolon. More than one keyword can be given by separating the keywords with semicolons (;). A topic cannot contain keywords unless it also has a topic title.</td></tr><tr><td>!</td><td>Specifies a Help macro. Windows Help executes the macro when the topic is displayed. The text parameter can be any Help macro.</td></tr></tbody></table> <p>If n is any letter (other than K), the footnote specifies an alternative keyword. Windows applications can search for topics having alternative keywords by using the <u>HELP_MULTIKEY</u> command with the <u>WinHelp</u> function.</p> | Value | Meaning | * | Specifies a build tag. The Microsoft Help Compiler uses build tags to determine whether it should include the topic in the Help file. The text parameter can be any combination of characters but must not contain spaces. Uppercase and lowercase characters are treated as equivalent characters (case-insensitive). If a topic has build-tag statements, they must be the first statements in the topic. The Microsoft Help Compiler checks a topic for build tags if the project file specifies a build expression using the <u>BUILD</u> option. | # | Specifies a context string. The text parameter can be any combination of letters and digits but must not contain spaces. Uppercase and lowercase characters are treated as equivalent characters (case-insensitive). The context string can be used with the <u>\v</u> statement in other topics to create links to this topic. | \$ | Specifies a topic title. Windows Help uses the topic title to identify the topic in the Search and History dialog boxes. The text parameter can be any combination of characters including spaces. | + | Specifies the browse-sequence identifier. Windows Help adds topics having an identifier to the browse sequence and allows users to view the topics by using the browse buttons. The text parameter can be a combination of letters and digits. Windows Help determines the order of topics in the browse sequence by sorting the identifier alphabetically. If two topics have the same identifier, Windows Help assumes that the topic that was compiled first is to be displayed first. Windows Help uses the browse sequence identifier only if the browse buttons have been enabled by using the <u>BrowseButtons</u> macro. | K | Specifies a keyword. Windows Help displays all keywords in the Help file in the Search dialog box and allows a user to choose a topic to view by choosing a keyword. The text parameter can be any combination of characters including spaces. If the first character is the letter K, it must be preceded with an extra space or a semicolon. More than one keyword can be given by separating the keywords with semicolons (;). A topic cannot contain keywords unless it also has a topic title. | ! | Specifies a Help macro. Windows Help executes the macro when the topic is displayed. The text parameter can be any Help macro. |
| Value     | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |                                                                                                                                                                                                                                                                                                                                 |    |                                                                                                                                                                                                    |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |   |                                                                                                                                |
| *         | Specifies a build tag. The Microsoft Help Compiler uses build tags to determine whether it should include the topic in the Help file. The text parameter can be any combination of characters but must not contain spaces. Uppercase and lowercase characters are treated as equivalent characters (case-insensitive). If a topic has build-tag statements, they must be the first statements in the topic. The Microsoft Help Compiler checks a topic for build tags if the project file specifies a build expression using the <u>BUILD</u> option.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |                                                                                                                                                                                                                                                                                                                                 |    |                                                                                                                                                                                                    |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |   |                                                                                                                                |
| #         | Specifies a context string. The text parameter can be any combination of letters and digits but must not contain spaces. Uppercase and lowercase characters are treated as equivalent characters (case-insensitive). The context string can be used with the <u>\v</u> statement in other topics to create links to this topic.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |                                                                                                                                                                                                                                                                                                                                 |    |                                                                                                                                                                                                    |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |   |                                                                                                                                |
| \$        | Specifies a topic title. Windows Help uses the topic title to identify the topic in the Search and History dialog boxes. The text parameter can be any combination of characters including spaces.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |                                                                                                                                                                                                                                                                                                                                 |    |                                                                                                                                                                                                    |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |   |                                                                                                                                |
| +         | Specifies the browse-sequence identifier. Windows Help adds topics having an identifier to the browse sequence and allows users to view the topics by using the browse buttons. The text parameter can be a combination of letters and digits. Windows Help determines the order of topics in the browse sequence by sorting the identifier alphabetically. If two topics have the same identifier, Windows Help assumes that the topic that was compiled first is to be displayed first. Windows Help uses the browse sequence identifier only if the browse buttons have been enabled by using the <u>BrowseButtons</u> macro.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |                                                                                                                                                                                                                                                                                                                                 |    |                                                                                                                                                                                                    |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |   |                                                                                                                                |
| K         | Specifies a keyword. Windows Help displays all keywords in the Help file in the Search dialog box and allows a user to choose a topic to view by choosing a keyword. The text parameter can be any combination of characters including spaces. If the first character is the letter K, it must be preceded with an extra space or a semicolon. More than one keyword can be given by separating the keywords with semicolons (;). A topic cannot contain keywords unless it also has a topic title.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |                                                                                                                                                                                                                                                                                                                                 |    |                                                                                                                                                                                                    |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |   |                                                                                                                                |
| !         | Specifies a Help macro. Windows Help executes the macro when the topic is displayed. The text parameter can be any Help macro.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |                                                                                                                                                                                                                                                                                                                                 |    |                                                                                                                                                                                                    |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |   |                                                                                                                                |
| text      | Specifies the build tag, context string, topic title, browse-sequence number, keyword, or macro associated with the footnote. This parameter depends on the footnote type as specified by the n parameter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |                                                                                                                                                                                                                                                                                                                                 |    |                                                                                                                                                                                                    |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |   |                                                                                                                                |

### Comments

Repetition of the footnote character, n, in the syntax is deliberate.

A topic can have more than one build-tag, context-string, keyword, and help-macro statement, but must not have more than one topic-title or browse-sequence-number statement.

In print-based documents, the \footnote statement creates a footnote. The footnote is anchored to the character immediately preceding the \footnote statement.

The characters in a context string must be alphanumeric and can include underscore characters ( \_ ) and periods ( . ).

The browse sequence string consists of a major sequence string and a minor sequence string, delimited by a colon:

`{+}{\footnote {+} major:minor}`

This syntax specifies disjoint sets of ordered browse sequences. The major sequence string determines which browse sequence a topic belongs to, while the minor sequence string determines its position. Minor sequence strings are sorted alphabetically, not numerically; to use numbers, they should be preceded with zeros so that they are all the same length. All topics with browse sequence strings that omit the major sequence string are placed on the same browse sequence.

A topic cannot have more than one build tag footnote. If a topic has a build tag footnote, it must be the first thing in that topic. The title, browse sequence, and macro must be in the first paragraph. Context strings and keywords may appear anywhere; if placed in the middle of a topic, jumps to that context string or keyword will bring you to the middle of that topic.

#### Example

The following example defines a topic titled "Short Topic". The context string "topic1" can be used to create links to this topic. The keywords "example topic" and "short topic" appear in the Search dialog box and can be used to choose the topic for viewing:

```
 ${\footnote Short Topic}
 #{\footnote topic1}
 K{\footnote example topic;short topic}
 This topic has a title, context string, and two keywords.
 \par
 \page
```

See Also

[\v](#)

## `\fs` RTF statement

`\fsn`

The `\fs` statement sets the size of the font. The new font size applies to all subsequent text up to the next `\plain` or `\fs` statement.

| Parameter      | Description                                     |
|----------------|-------------------------------------------------|
| <code>n</code> | Specifies the size of the font, in half points. |

### Comments

The `\fs` statement does not set the font face; use the `\f` statement instead.

No `\plain` or `\fs` statement is required if the `\fs` statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

If the `\fs` statement is not given, the default font size is 24.

### Example

The following example sets the size of the font to 10 points:

```
{\fs20 This line is in 10 point type.}  
\par
```

### See Also

[\plain](#), [\f](#)

## \ RTF statement

### \hh

The \ statement converts the specified hexadecimal number into a character value and inserts the value into the Help file. The appearance of the character when displayed depends on the character set specified for the Help file.

| Parameter | Description                              |
|-----------|------------------------------------------|
| hh        | Specifies a two-digit hexadecimal value. |

### Comments

Since the Microsoft Help Compiler does not accept character values greater than 127, the \ statement is the only way to insert such character values into the Help file.

### Example

The following example inserts a trademark in a Help file that uses the \ansi statement to set the character set:

```
ABC\'99 is a trademark of the ABC Product Corporation.
```

### See Also

\ansi, \mac, \pc

## `\i` RTF statement

`\i`

The `\i` statement starts italic text. The statement applies to all subsequent text up to the next `\plain` or `\i0` statement.

### Comments

No `\plain` or `\i0` statement is required if the `\i` statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

### Example

The following example sets "not" to italic:

```
You must {\i not} save the file without first setting the  
Auto option.
```

### See Also

`\b`, `\plain`, `\scaps`



## `\intbl` RTF statement (3.1)

### `\intbl`

The `\intbl` statement marks subsequent paragraphs as part of a table. The statement applies to all subsequent paragraphs up to the next `\row` statement.

### Comments

This statement was first supported in Microsoft Help Compiler version 3.1.

### Example

The following example creates a three-column table having two rows:

```
\cellx1440\cellx2880\cellx4320
\intbl
Row 1 Column 1\cell
Row 1 Column 2\cell
Row 1 Column 3\cell \row
\intbl
Row 2 Column 1\cell
Row 2 Column 2\cell
Row 2 Column 3\cell \row \pard
```

### See Also

`\cell`, `\cellx`, `\row`, `\trgaph`, `\trleft`, `\trowd`

## `\keep` RTF statement

### `\keep`

The `\keep` statement prevents Windows Help from wrapping text to fit the Help window. The statement applies to all subsequent paragraphs up to the next `\pard` statement.

### Comments

If the text in a paragraph exceeds the width of the Help window, Help displays a horizontal scroll bar.

In print-based documents, the `\keep` statement keeps paragraphs intact.

### See Also

`\keepn`, `\line`

## `\keepn` RTF statement

### `\keepn`

The `\keepn` statement creates a non-scrolling region at the top of the Help window for the given topic. The `\keepn` statement applies to all subsequent paragraphs up to the next `\pard` statement. All paragraphs with this paragraph property are placed in the non-scrolling region.

### Comments

If a `\keepn` statement is used in a topic, it must be applied to the first paragraph in the topic (and subsequent paragraphs as needed). The Help compiler displays an error message and does not create a non-scrolling region if paragraphs are given before the `\keepn` statement. Only one non-scrolling region per topic is allowed.

Windows Help formats, aligns, and wraps text in the non-scrolling region just as it does in the rest of the topic. It separates the non-scrolling region from the rest of the Help window with a horizontal bar.

Windows Help sets the height of the non-scrolling region so that all paragraphs in the region can be viewed if the help window is large enough. If the window is smaller than the non-scrolling region, the user will be unable to view the rest of the topic. For this reason, the non-scrolling region is typically reserved for a single line of text specifying the name or title of the topic.

In print-based documents, the `\keepn` statement keeps the subsequent paragraph with the paragraph that follows it.

### See Also

`\keep`, `\page`

## \li RTF statement

### \lin

The \li statement sets the left indent for the paragraph. The indent applies to all subsequent paragraphs up to the next \pard or \li statement.

| Parameter | Description                                                                   |
|-----------|-------------------------------------------------------------------------------|
| n         | Specifies the indent, in twips. The value can be either positive or negative. |

### Comments

If the \li statement is not given, the left indent is zero by default. Windows Help automatically provides a small left margin so that if no indent is specified the text does not start immediately at the left edge of the Help window.

Specifying a negative left indent moves the starting point for a line of text to the left of the default left margin. If the negative indent is large enough, the start of the text may be clipped by the left edge of the help window.

### Example

The following example uses the left indent and a tab stop to make a bulleted list. In this example, font number 0 is assumed to be the Symbol font:

Use the Auto command to:

```
\par
\tx360\li360\fi-360
{\f0\B7}
\tab
Save files automatically
\par
{\f0\B7}
\tab
Prevent overwriting existing files
\par
{\f0\B7}
\tab
Create automatic backup files
\par \pard
```

### See Also

\fi, \pard, \ri

\line RTF statement

\line

The \line statement breaks the current line without ending the paragraph. Subsequent text starts on the next line and is aligned and indented according to the current paragraph properties.

See Also

\par

`\mac` RTF statement

`\mac`

The `\mac` statement sets the Apple Macintosh character set.

See Also

`\ansi`, `\pc`

## `\page` RTF statement

`\page`

The `\page` statement marks the end of a topic.

### Comments

In a print-based document, the `\page` statement creates a page break.

### Example

The following example shows a complete topic:

```
${\footnote Short Topic}
```

```
#{\footnote short_topic}
```

Most topics in a topic file consist of topic-title and context-string statements followed by the topic text. Every topic ends with a `{\b \page}` statement.

```
\par
```

```
\page
```

See Also

[\par](#)

## `\par` RTF statement

`\par`

The `\par` statement marks the end of a paragraph. The statement ends the current line of text and moves the current position to the left margin and down by the current line-spacing and space-after-paragraph values.

### Comments

The first line of text after a `\par`, `\page`, or `\sect` statement marks the start of a paragraph. When a paragraph starts, the current position is moved down by the current space-before-paragraph value. Subsequent text is formatted using the current text alignment, line spacing, and left, right, and first-line indents.

### Example

The following example has three paragraphs:

```
\ql  
This paragraph is left-aligned.  
\par \pard  
\qc  
This paragraph is centered.  
\par \pard  
\qr  
This paragraph is right-aligned.  
\par
```

### See Also

[`\line`](#), [`\pard`](#), [`\sect`](#)



## `\pard` RTF statement

`\pard`

The `\pard` statement restores all paragraph properties to default values.

### Comments

If the `\pard` statement appears anywhere before the end of a paragraph (that is, before the `\par` statement), the default properties apply to the entire paragraph.

The default paragraph properties are as follows:

| Property          | Default           |
|-------------------|-------------------|
| Alignment         | Left-aligned      |
| First-line indent | 0                 |
| Left indent       | 0                 |
| Right indent      | 0                 |
| Space before      | 0                 |
| Space after       | 0                 |
| Line spacing      | Tallest character |
| Tab stops         | None              |
| Borders           | None              |
| Border style      | Single-width      |

See Also

[`\par`](#)

`\pc` RTF statement

`\pc`

The `\pc` statement sets the OEM character set (also known as code page 437).

See Also

`\ansi`, `\mac`

## `\pich` RTF statement

### `\pichn`

The `\pich` statement specifies the height of the picture. This statement must be used in conjunction with a `\pict` statement.

| Parameter | Description                                                                                                                                                                      |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| n         | Specifies the height of the picture, in twips or pixels, depending on the picture type. If the picture is a metafile, the width is in twips; otherwise, the width is, in pixels. |

See Also

[`\pict`](#), [`\picw`](#)

## `\pichgoal` RTF statement

### `\pichgoaln`

The `\pichgoal` statement specifies the desired height of a picture. If necessary, Windows Help stretches or compresses the picture to match the requested height. This statement must be used in conjunction with a `\pict` statement.

| Parameter | Description                             |
|-----------|-----------------------------------------|
| n         | Specifies the desired height, in twips. |

### Comments

The `\pichgoal` statement is not supported for metafiles. Applications should use the `\pich` statement, instead.

### See Also

`\pich`, `\pict`, `\picwgoal`

## `\picscalex` RTF statement

### `\picscalexn`

The `\picscalex` statement specifies the horizontal scaling value. This statement must be used in conjunction with a `\pict` statement.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

|                |  |
|----------------|--|
| <code>n</code> |  |
|----------------|--|

|  |                                                                                                                     |
|--|---------------------------------------------------------------------------------------------------------------------|
|  | Specifies the scaling value as a percentage. If this value is greater than 100, the bitmap or metafile is enlarged. |
|--|---------------------------------------------------------------------------------------------------------------------|

### Comments

If the `\picscalex` statement is not given, the default scaling value is 100.

### See Also

`\pict`, `\picscaley`

## `\picscaley` RTF statement

### `\picscaleyn`

The `\picscaley` statement specifies the vertical scaling value. This statement must be used in conjunction with a `\pict` statement.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

|                |  |
|----------------|--|
| <code>n</code> |  |
|----------------|--|

|  |                                                                                                                     |
|--|---------------------------------------------------------------------------------------------------------------------|
|  | Specifies the scaling value as a percentage. If this value is greater than 100, the bitmap or metafile is enlarged. |
|--|---------------------------------------------------------------------------------------------------------------------|

### Comments

If the `\picscaley` statement is not given, the default scaling value is 100.

### See Also

`\pict`, `\picscalex`

## `\pict` RTF statement

`\pict` picture-statements picture-data

The `\pict` statement creates a picture. A picture consists of hexadecimal or binary data representing a bitmap or metafile.

| Parameter                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|------------|-----------------------|-----------------------------|-------------------------|-------------------------------|--------------------|------------------------------|--------------------|-------------------------------|------------------------|--------------------------------------|------------------------|---------------------------------------|-------------------------|-----------------------------------------|-------------------------|---------------------------------------|----------------------------|-----------------------------------------|-------------------------|---------------------------------|-----------------------------|---------------------------------------|-------------------|--------------------------------|
| picture-statements          | Specifies one or more statements defining the type of picture, the dimensions of the picture, and the format of the picture data. It can be a combination of the following statements: <table><thead><tr><th>Statement</th><th>Descripton</th></tr></thead><tbody><tr><td><code>\wbitmap</code></td><td>Specifies a Windows bitmap.</td></tr><tr><td><code>\wmetafile</code></td><td>Specifies a Windows metafile.</td></tr><tr><td><code>\picw</code></td><td>Specifies the picture width.</td></tr><tr><td><code>\pich</code></td><td>Specifies the picture height.</td></tr><tr><td><code>\picwgoal</code></td><td>Specifies the desired picture width.</td></tr><tr><td><code>\pichgoal</code></td><td>Specifies the desired picture height.</td></tr><tr><td><code>\picscalex</code></td><td>Specifies the horizontal scaling value.</td></tr><tr><td><code>\picscaley</code></td><td>Specifies the vertical scaling value.</td></tr><tr><td><code>\wbmbitspixel</code></td><td>Specifies the number of bits per pixel.</td></tr><tr><td><code>\wbmplanes</code></td><td>Specifies the number of planes.</td></tr><tr><td><code>\wbmwidthbytes</code></td><td>Specifies the bitmap width, in bytes.</td></tr><tr><td><code>\bin</code></td><td>Specifies binary picture data.</td></tr></tbody></table> | Statement | Descripton | <code>\wbitmap</code> | Specifies a Windows bitmap. | <code>\wmetafile</code> | Specifies a Windows metafile. | <code>\picw</code> | Specifies the picture width. | <code>\pich</code> | Specifies the picture height. | <code>\picwgoal</code> | Specifies the desired picture width. | <code>\pichgoal</code> | Specifies the desired picture height. | <code>\picscalex</code> | Specifies the horizontal scaling value. | <code>\picscaley</code> | Specifies the vertical scaling value. | <code>\wbmbitspixel</code> | Specifies the number of bits per pixel. | <code>\wbmplanes</code> | Specifies the number of planes. | <code>\wbmwidthbytes</code> | Specifies the bitmap width, in bytes. | <code>\bin</code> | Specifies binary picture data. |
| Statement                   | Descripton                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
| <code>\wbitmap</code>       | Specifies a Windows bitmap.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
| <code>\wmetafile</code>     | Specifies a Windows metafile.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
| <code>\picw</code>          | Specifies the picture width.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
| <code>\pich</code>          | Specifies the picture height.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
| <code>\picwgoal</code>      | Specifies the desired picture width.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
| <code>\pichgoal</code>      | Specifies the desired picture height.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
| <code>\picscalex</code>     | Specifies the horizontal scaling value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
| <code>\picscaley</code>     | Specifies the vertical scaling value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
| <code>\wbmbitspixel</code>  | Specifies the number of bits per pixel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
| <code>\wbmplanes</code>     | Specifies the number of planes.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
| <code>\wbmwidthbytes</code> | Specifies the bitmap width, in bytes.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
| <code>\bin</code>           | Specifies binary picture data.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |
| picture-data                | Specifies hexadecimal or binary data representing the picture. The picture data follows the last picture statement.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |           |            |                       |                             |                         |                               |                    |                              |                    |                               |                        |                                      |                        |                                       |                         |                                         |                         |                                       |                            |                                         |                         |                                 |                             |                                       |                   |                                |

### Comments

If a data format is not specified, the default format is hexadecimal.

### See Also

`\bin`, `\pich`, `\pichgoal`, `\picscalex`, `\picscaley`, `\picw`, `\picwgoal`, `\wbitmap`, `\wbmbitspixel`, `\wbmplanes`, `\wbmwidthbytes`, `\wmetafile`

## \picw RTF statement

\picwn

The \picw statement specifies the width of the picture. This statement must be used in conjunction with a \pict statement.

| Parameter | Description                                                                                                                                                                    |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| n         | Specifies the width of the picture, in twips or pixels, depending on the picture type. If the picture is a metafile, the width is in twips; otherwise, the width is in pixels. |

See Also

\pict, \pich



## \picwgoal RTF statement

\picwgoaln

The \picwgoal statement specifies the desired width of the picture, in twips. If necessary, Windows Help stretches or compresses the picture to match the requested height. This statement must be used in conjunction with a \pict statement.

| Parameter | Description                            |
|-----------|----------------------------------------|
| n         | Specifies the desired width, in twips. |

### Comments

The \picwgoal statement is not supported for metafiles. Applications should use the \picw statement, instead.

### See Also

\pict, \picw, \pichgoal

## `\plain` RTF statement

### `\plain`

The `\plain` statement restores the character properties to default values.

#### Comments

The default character properties are as follows:

| Property   | Default |
|------------|---------|
| Bold       | Off     |
| Italic     | Off     |
| Small caps | Off     |
| Font       | 0       |
| Font size  | 24      |

#### See Also

`\b`, `\i`, `\scaps`, `\f`, `\fs`

## `\qc` RTF statement

`\qc`

The `\qc` statement centers text between the current left and right indents. The statement applies to subsequent paragraphs up to the next `\pard` statement or text-alignment statement.

### Comments

If a `\ql`, `\qr`, or `\qc` statement is not given, the text is left-aligned by default.

### See Also

`\pard`, `\ql`, `\qr`

## \ql RTF statement

\ql

The \ql statement aligns text along the left indent. The statement applies to subsequent paragraphs up to the next \pard statement or text-alignment statement.

### Comments

If a \ql, \qr, or \qc statement is not given, the text is left-aligned by default.

### See Also

\pard, \qc, \qr

## \qr RTF statement

\qr

The \qr statement aligns text along the right indent. The statement applies to subsequent paragraphs up to the next \pard statement or text-alignment statement.

### Comments

If a \ql, \qr, or \qc statement is not given, the text is left-aligned by default.

### See Also

\pard, \qc, \ql

## `\ri` RTF statement

`\rin`

The `\ri` statement sets the right indent for the paragraph. The indent applies to all subsequent paragraphs up to the next `\pard` or `\ri` statement.

| Parameter      | Description                                                                   |
|----------------|-------------------------------------------------------------------------------|
| <code>n</code> | Specifies the right indent, in twips. It can be a positive or negative value. |

### Comments

If the `\ri` statement is not given, the right indent is zero by default. Windows Help automatically provides a small right margin so that when no right indent is specified, the text does not end abruptly at the right edge of the Help window.

Windows Help never displays less than one word for each line in a paragraph even if the right indent is greater than the width of the window.

### Example

In the following example, the right and left indents are set to one inch and the subsequent text is centered between the indents:

```
\li1440\ri1440\qc  
Microsoft Windows Help\line  
Sample File\line
```

### See Also

[\li](#), [\pard](#)

## `\row` RTF statement

`\row`

The `\row` statement marks the end of a table row. The statement ends the current row and begins a new row by moving down past the end of the longest cell in the row. The next `\cell` statement specifies the text of the leftmost cell in the next row.

### Comments

This statement was first supported in the Microsoft Help Compiler version 3.1.

### Example

The following example creates a table having four rows and two columns:

```
\cellx2880\cellx5760
\intbl
Row 1, Column 1\cell
Row 1, Column 2\cell \row
\intbl
Row 2, Column 1\cell
Row 2, Column 2\cell \row
\intbl
Row 3, Column 1\cell
Row 3, Column 2\cell \row
\intbl
Row 4, Column 1\cell
Row 4, Column 2\cell \row
\par \pard
```

### See Also

`\cell`, `\cellx`, `\intbl`

## `\rtf` RTF statement

### `\rtfn`

The `\rtf` statement identifies the file as a rich-text format (RTF) file and specifies the version of the RTF standard used.

| Parameter | Description                                                                                                            |
|-----------|------------------------------------------------------------------------------------------------------------------------|
| n         | Specifies the version of the RTF standard used. For the Microsoft Help Compiler version 3.1, this parameter must be 1. |

### Comments

The `\rtf` statement must follow the first open brace in the Help file. A statement specifying the character set for the file must also follow the `\rtf` statement.

### See Also

[`\ansi`](#)



## `\sa` RTF statement

`\san`

The `\sa` statement sets the amount of vertical spacing after a paragraph. The vertical space applies to all subsequent paragraphs up to the next `\pard` or `\sa` statement.

| Parameter      | Description                                         |
|----------------|-----------------------------------------------------|
| <code>n</code> | Specifies the amount of vertical spacing, in twips. |

### Comments

If the `\sa` statement is not given, the vertical spacing after a paragraph is zero by default.

See Also

`\sb`, `\pard`

## `\sb` RTF statement

`\sbn`

The `\sb` statement sets the amount of vertical spacing before the paragraph. The vertical space applies to all subsequent paragraphs up to the next `\pard` statement or `\sb` statement.

| Parameter      | Description                                         |
|----------------|-----------------------------------------------------|
| <code>n</code> | Specifies the amount of vertical spacing, in twips. |

### Comments

If the `\sb` statement is not given, the vertical spacing before the paragraph is zero by default.

See Also

[\sa](#), [\pard](#)

## `\scaps` RTF statement

### `\scaps`

The `\scaps` statement starts small-capital text. The statement converts all subsequent lowercase letters to uppercase before displaying the text. This statement applies to all subsequent text up to the next `\plain` or `\scaps0` statement.

### Comments

The `\scaps` statement does not affect uppercase letters.

No `\plain` or `\scaps0` statement is required if the `\scaps` statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

The `\scaps` statement does not reduce the point size of the text. To reduce point size, the `\fs` statement must be used.

### Example

The following example displays the key name ENTER in small capitals:

Press the `{\scaps enter}` key to complete the action.

### See Also

`\fs`, `\plain`

`\sect` RTF statement

`\sect`

The `\sect` statement marks the end of a section and paragraph.

See Also

`\par`

## `\sl` RTF statement

### `\sln`

The `\sl` statement sets the amount of vertical space between lines in a paragraph. The vertical space applies to all subsequent paragraphs up to the next `\pard` or `\sl` statement.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| n         | Specifies the amount of vertical spacing, in twips. If this parameter is a positive value, Windows Help uses this value if it is greater than the tallest character. Otherwise, Windows Help uses the height of the tallest character as the line spacing. If this parameter is a negative value, Windows Help uses the absolute value of the number even if the tallest character is taller. |

### Comments

If the `\sl` statement is not given, Windows Help automatically sets the line spacing by using the tallest character in the line.

### See Also

`\pard`

## \strike RTF statement

### \strike

The \strike statement creates a hot spot. The statement is used in conjunction with a \v statement to create a link to another topic. When the user chooses a hot spot, Windows Help displays the associated topic in the Help window.

The \strike statement applies to all subsequent text up to the next \plain or \strike0 statement.

### Comments

No \plain or \strike0 statement is required if the \strike statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

The \strike statement creates the same type of hot spot as the \uldb statement.

In print-based documents, or whenever it is not followed by \v, the \strike statement creates strikeout text.

### Example

The following example creates a hot spot for a topic. When displayed, the hot-spot text, "Hot Spot," is green and has a solid line under it:

```
{\strike Hot Spot}{\v Topic}
```

### See Also

\ul, \uldb, \v

`\tab` RTF statement

`\tab`

The `\tab` statement inserts a tab character (ASCII character code 9).

Comments

The tab character (ASCII character code 9) has the same effect as the `\tab` statement.

See Also

[\tqc](#), [\tqr](#), [\tx](#)

## `\tqc` RTF statement

`\tqc`

The `\tqc` statement is used with the `\tx` statement to create a tab stop where text is centered. For example, the following statement creates a centered tab stop at 2880 twips:

```
\tqc\tx2880
```

See Also

[\tab](#), [\tqr](#), [\tx](#)



## `\tqr` RTF statement

`\tqr`

The `\tqr` statement is used with the `\tx` statement to create a tab stop where text right-justified. For example, the following statement creates a right-justified tab stop at 2880 twips:

```
\tqr\tx2880
```

See Also

`\tab`, `\tqc`, `\tx`

## `\trgaph` RTF statement (3.1)

### `\trgaphn`

The `\trgaph` statement specifies the amount of space between text in adjacent cells in a table. For each cell in the table, Windows Help uses the space to calculate the cell's left and right margins. It then uses the margins to align and wrap the text in the cell. Windows Help applies the same margin widths to each cell ensuring that paragraphs in adjacent cells have the specified space between them.

The `\trgaph` statement applies to cells in all subsequent rows of a table up to the next `\trowd` statement.

| Parameter      | Description                                                                                                                                                                                           |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>n</code> | Specifies the space, in twips, between text in adjacent cells. If this parameter exceeds the actual width of the cell, the left and right margins are assumed to be at the same position in the cell. |

### Comments

The width of the left margin in the first cell is always equal to the space specified by this statement. The `\trleft` statement is typically used to move the left margin to a position similar to the left margins in all other cells.

This statement was first supported in the Microsoft Help Compiler version 3.1.

### Example

The following example creates a three-column table with one-quarter inch space between the text in the columns:

```
\trgaph360 \cellx1440\cellx2880\cellx4320
\intbl
Row 1 Column 1\cell
Row 1 Column 2\cell
Row 1 Column 3\cell \row
\intbl
Row 2 Column 1\cell
Row 2 Column 2\cell
Row 2 Column 3\cell \row \pard
```

### See Also

`\cell`, `\cellx`, `\intbl`, `\row`, `\trleft`, `\trowd`

## `\trleft` RTF statement

### `\trleftn`

The `\trleft` statement sets the position of the left margin for the first (leftmost) cell in a row of a table. This statement applies to the first cell in all subsequent rows of the table up to the next `\trowd` statement.

| Parameter      | Description                                                                                                                                                                                                  |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>n</code> | Specifies the relative position, in twips, of the left margin. This parameter can be a positive or negative number. The final position of the left margin is the sum of the current position and this value. |

### Comments

This statement was first supported in the Microsoft Help Compiler version 3.1.

### Example

The following example creates a three-column table with one-quarter inch space between the text in the columns. The left margin in the first cell is flush with the left margin of the Help window:

```
\trgaph360\trleft-360 \cellx1440\cellx2880\cellx4320
\intbl
Row 1 Column 1\cell
Row 1 Column 2\cell
Row 1 Column 3\cell \row
\intbl
Row 2 Column 1\cell
Row 2 Column 2\cell
Row 2 Column 3\cell \row \pard
```

### See Also

[\cell](#), [\cellx](#), [\intbl](#), [\row](#), [\trgaph](#), [\trowd](#)

## `\trowd` RTF statement

`\trowd`

The `\trowd` statement sets default margins and cell positions for subsequent rows in a table.

### Comments

This statement was first supported in the Microsoft Help Compiler version 3.1.

### See Also

[`\cell`](#), [`\cellx`](#), [`\intbl`](#), [`\row`](#), [`\trgaph`](#), [`\trleft`](#)

## `\trqc` RTF statement

### `\trqc`

The `\trqc` statement directs Windows Help to dynamically adjust the width of table columns to fit in the current window.

### Comments

In a print-based document, the `\trqc` statement centers a table row with respect to its containing column.

Windows Help will not resize a table to smaller than the widths specified in the `\trqc` statement. Therefore, the table should be created in the smallest size in which it would ever be displayed. All columns in the table are sized proportionally.

This statement was first supported in the Microsoft Help Compiler version 3.1.

### See Also

[`\trowd`](#), [`\trql`](#)

## `\trql` RTF statement

`\trql`

The `\trql` statement aligns the text in each cell of a table row to the left.

### Comments

This statement was first supported in the Microsoft Help Compiler version 3.1.

### See Also

[`\trowd`](#), [`\trqc`](#)

## `\tx` RTF statement

`\txn`

The `\tx` statement sets the position of a tab stop. The position is relative to the left margin of the Help window. A tab stop applies to all subsequent paragraphs up the next `\pard` statement.

| Parameter      | Description                                |
|----------------|--------------------------------------------|
| <code>n</code> | Specifies the tab stop position, in twips. |

### Comments

If the `\tx` statement is not given, tab stops are set at every one-half inch by default.

### See Also

`\tab`, `\tqc`, `\tqr`

## `\ul` RTF statement

### `\ul`

The `\ul` statement creates a link to a pop-up topic. The statement is used in conjunction with a `\v` statement to create a link to another topic. When the user chooses the link, Windows Help displays the associated topic in a pop-up window.

The `\ul` statement applies to all subsequent text up to the next `\plain` or `\ul0` statement.

### Comments

No `\plain` or `\ul0` statement is required if the `\ul` statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

In print-base documents, or whenever it is not followed by `\v`, the `\ul` statement creates a continuous underline.

### Example

The following example creates a pop-up link for a topic. When displayed, the link text, "Popup Link," is green and has a dotted line under it:

```
{\ul Popup Link}{\v PopupTopic}
```

### See Also

`\strike`, `\uldb`, `\v`



## \uldb RTF statement

### \uldb

The \uldb statement creates a hot spot. This statement is used in conjunction with a \v statement to create a link to another topic. When the user chooses a hot spot, Windows Help displays the associated topic in the Help window.

The \uldb statement applies to all subsequent text up to the next \plain or \uldb0 statement.

### Comments

No \plain or \uldb0 statement is required if the \uldb statement and subsequent text are enclosed in braces. Braces limit the scope of a character property statement to just the enclosed text.

The \uldb statement creates the same type of hot spot as the \strike statement.

### Example

The following example creates a hot spot for a topic. When displayed, the hot-spot text, "Hot Spot," is green and has a solid line under it:

```
{\uldb Hot Spot}{\v Topic}
```

### See Also

\strike, \ul, \v

## \v RTF statement

{\v context-string}

The \v statement creates a link to the topic having the specified context string. The \v statement is used in conjunction with the [\strike](#), [\ul](#), and [\uldb](#) statements to create hot spots and links to topics.

| Parameter      | Description                                                                                                                                                                                                                                  |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| context-string | Specifies the context string of a topic in the Help file. The string can be any combination of characters, except spaces, and must also be specified in a context-string <a href="#">\footnote</a> statement in some topic in the Help file. |

### Comments

If the context string is preceded by a percent sign (%), Windows Help displays the associated hot spot or link without applying the standard underline and color. If the context string is preceded by an asterisk (\*), Windows Help displays the associated hot spot or link with an underline but without applying the standard color.

In print-based documents, the \v statement creates hidden text.

For links or hot spots, the syntax of the \v statement is as follows:

```
[%|*] context [>secondary-window] [@filename]
```

In this syntax, secondary-window is the name of the secondary window to jump to. When the secondary window is not specified, the jump is to the same window as the current help topic is using. To jump to the main window, specify "main" for this parameter. This parameter may not be used with pop-up windows.

The filename parameter specifies a jump to a topic in a different help file.

For a macro hotspot, the syntax of the \v statement is as follows:

```
[%|*] ! macro [;macro][;...]
```

### Example

The following example creates a hot spot for the topic having the context string "Topic". Windows Help applies an underline and the color green the text "Hot Spot" when it displays the topic:

```
{\uldb Hot Spot}{\v Topic}
```

### See Also

[\footnote](#), [\strike](#), [\ul](#), [\uldb](#)

## \wbitmap RTF statement

### \wbitmapn

The \wbitmap statement sets the picture type to Windows bitmap. This statement must be used in conjunction with a \pict statement.

| Parameter | Description                                                             |
|-----------|-------------------------------------------------------------------------|
| n         | Specifies the bitmap type. This parameter is zero for a logical bitmap. |

### Comments

The \wbitmap statement is optional; if a \wmetafile statement is not specified, the picture is assumed to be a Windows bitmap.

### Example

The following example creates a 32-by-8 pixel monochrome bitmap:

```
{  
\pict \wbitmap0\wbmbitspixel1\wbmplanes1\wbmwidthbytes4\picw32\pich8  
3FFFFFFC  
F3FFFFFFCF  
FF3FFCF  
FFF3CFFF  
FFFC3FFF  
FFCFF3FF  
FCFFFF3F  
CFFFFFFF3  
}
```

### See Also

bmc, bml, bmr, \pict, \wmetafile

## \wbmbitspixel RTF statement

\wbmbitspixeln

The \wbmbitspixel statement specifies the number of consecutive bits in the bitmap data that represent a single pixel. This statement must be used in conjunction with the \pict statement.

| Parameter | Description                             |
|-----------|-----------------------------------------|
| n         | Specifies the number of bits per pixel. |

### Comments

If the \wbmbitspixel statement is not given, the default bits per pixel value is 1.

### See Also

\pict, \wbitmap, \wbmplanes

## \wbmplanes RTF statement

\wbmplanesn

The \wbmplanes statement specifies the number of color planes in the bitmap data. This statement must be used in conjunction with a \pict statement.

| Parameter | Description                            |
|-----------|----------------------------------------|
| n         | Specifies the number of bitmap planes. |

### Comments

If the \wbmplanes statement is not given, the default number of planes is 1.

### See Also

\pict, \wbitmap, \wbmbitspixel

## `\wbmwidthbytes` RTF statement

`\wbmwidthbytesn`

The `\wbmwidthbytes` statement specifies the number of bytes in each scan line of the bitmap data. This statement must be used in conjunction with the `\pict` statement.

| Parameter | Description                                  |
|-----------|----------------------------------------------|
| n         | Specifies the width of the bitmap, in bytes. |

See Also

`\pict`, `\wbitmap`

## \wmetafile RTF statement

### \wmetafilen

The \wmetafile statement sets the picture type to a Windows metafile. This statement must be used in conjunction with the \pict statement.

| Parameter | Description                                            |
|-----------|--------------------------------------------------------|
| n         | Specifies the metafile type. This parameter must be 8. |

### Comments

Windows Help expects the hexadecimal data associated with the picture to represent a valid Windows metafile. By default, Windows Help sets the MM\_ANISOTROPIC mapping mode prior to displaying the metafile. To ensure that the picture is displayed correctly, the metafile data must either set the window origin and extents by using the SetWindowOrg and SetWindowExt records or set another mapping mode by using the SetMapMode record.

### Example

The following example creates a picture using a metafile:

```
{\pict\wmetafile8\picw2880\pich2880
01000900000034f00000000200090000000000
0500000000b020000000000500000000c026400
64000900000001d066200ff00640064000000
0000080000000fa0200000200000000000000
040000002d01000005000000140200000000
05000000013026400640005000000014020000
640005000000013026400000008000000fa02
000000000000000000000040000002d010100
04000000f001000003000000000004e0dff00
87002000000500000020000000000000000000}
\par }
```

### See Also

bmc, bml, bmr, \pict, \wbimap

## AdjustWindowRect (2.x)

```
void AdjustWindowRect(lprc, dwStyle, fMenu)
RECT FAR* lprc;          /* address of client-rectangle structure */
DWORD dwStyle;          /* window styles */
BOOL fMenu;             /* menu-present flag */
```

The AdjustWindowRect function computes the required size of the window rectangle based on the desired client-rectangle size. The window rectangle can then be passed to the [CreateWindow](#) function to create a window whose client area is the desired size.

| Parameter | Description                                                                                       |
|-----------|---------------------------------------------------------------------------------------------------|
| lprc      | Points to a <a href="#">RECT</a> structure that contains the coordinates of the client rectangle. |
| dwStyle   | Specifies the window styles of the window whose client rectangle is to be converted.              |
| fMenu     | Specifies whether the window has a menu.                                                          |

### Returns

This function does not return a value.

### Comments

A client rectangle is the smallest rectangle that completely encloses a client area. A window rectangle is the smallest rectangle that completely encloses the window.

AdjustWindowRect does not take titles and borders into account when computing the size of the client area. For window styles that include titles and borders, applications must add the title and border sizes after calling AdjustWindowRect. This function also does not take the extra rows into account when a menu bar wraps to two or more rows.

### See Also

[AdjustWindowRectEx](#), [CreateWindowEx](#)



### AdjustWindowRectEx (3.0)

```
void AdjustWindowRectEx(lprc, dwStyle, fMenu, dwExStyle)
RECT FAR* lprc;          /* address of client-rectangle structure */
DWORD dwStyle;          /* window styles */
BOOL fMenu;             /* menu-present flag */
DWORD dwExStyle;        /* extended style */
```

The AdjustWindowRectEx function computes the required size of the rectangle of a window with extended style based on the desired client-rectangle size. The window rectangle can then be passed to the [CreateWindowEx](#) function to create a window whose client area is the desired size.

| Parameter | Description                                                                                       |
|-----------|---------------------------------------------------------------------------------------------------|
| lprc      | Points to a <a href="#">RECT</a> structure that contains the coordinates of the client rectangle. |
| dwStyle   | Specifies the window styles of the window whose client rectangle is to be converted.              |
| fMenu     | Specifies whether the window has a menu.                                                          |
| dwExStyle | Specifies the extended style of the window being created.                                         |

#### Returns

This function does not return a value.

#### Comments

A client rectangle is the smallest rectangle that completely encloses a client area. A window rectangle is the smallest rectangle that completely encloses the window.

AdjustWindowRectEx does not take titles and borders into account when computing the size of the client area. For window styles that include titles and borders, applications must add the title and border sizes after calling AdjustWindowRectEx. This function also does not take the extra rows into account when a menu bar wraps to two or more rows.

#### See Also

[AdjustWindowRect](#), [CreateWindowEx](#)

## AnsiLower (2.x)

LPSTR AnsiLower(lpsz)  
LPSTR lpsz; /\* address of string, or specific character \*/

The AnsiLower function converts a character string to lowercase.

| Parameter | Description                                                                                                                                                                                                     |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpsz      | Points to a null-terminated string or specifies a single character. If the high-order word of this parameter is zero, the low-order byte of the low-order word must contain a single character to be converted. |

### Returns

The return value points to a converted character string if the function is successful. Otherwise, the return value is a 32-bit value that contains the converted character in the low-order byte of the low-order word.

### Comments

The conversion is made by the language driver for the current language (the one selected by the user at setup or by using Control Panel). If no language driver has been selected, Windows uses an internal function.

### Example

The following example uses the AnsiLower function to convert two strings to lowercase for a non– case-sensitive comparison:

```
/*  
 * Convert the target string to lowercase, and then  
 * convert the subject string one character at a time.  
 */  
  
AnsiLower(pszTarget);  
while (*pszTarget != '\0') {  
    if (*pszTarget != (char) (DWORD) AnsiLower(  
        MAKELP(0, *pszSubject)))  
        return FALSE;  
    pszTarget = AnsiNext(pszTarget);  
    pszSubject = AnsiNext(pszSubject);  
}
```

### See Also

[AnsiLowerBuff](#), [AnsiNext](#), [AnsiUpper](#)

## AnsiLowerBuff (3.0)

```
UINT AnsiLowerBuff(lpszString, cbString)
LPSTR lpszString;          /* address of string to convert */
UINT cbString;             /* length of string*/
```

The AnsiLowerBuff function converts a character string in a buffer to lowercase.

| Parameter  | Description                                                                                                                          |
|------------|--------------------------------------------------------------------------------------------------------------------------------------|
| lpszString | Points to a buffer containing one or more characters.                                                                                |
| cbString   | Specifies the number of bytes in the buffer identified by the lpszString parameter. If cbString is zero, the length is 64K (65,536). |

### Returns

The return value specifies the length of the converted string if the function is successful. Otherwise, it is zero.

### Comments

The language driver makes the conversion for the current language (the one selected by the user at setup or by using Control Panel). If no language driver has been selected, Windows uses an internal function.

### Example

The following example uses the AnsiLowerBuff function to convert two strings to lowercase for a non- case-sensitive comparison:

```
AnsiLowerBuff(pszSubject, (UINT) strlen(pszSubject));
AnsiLowerBuff(pszTarget, (UINT) strlen(pszTarget));

while (*pszTarget != '\0') {
    if (*pszTarget != *pszSubject)
        return FALSE;
    pszTarget = AnsiNext(pszTarget);
    pszSubject = AnsiNext(pszSubject);
}
```

### See Also

AnsiLower, AnsiUpper

## AnsiNext (2.x)

```
LPSTR AnsiNext(lpchCurrentChar)
LPCSTR lpchCurrentChar;      /* address of current character */
```

The AnsiNext function moves to the next character in a string.

| Parameter       | Description                                        |
|-----------------|----------------------------------------------------|
| lpchCurrentChar | Points to a character in a null-terminated string. |

### Returns

The return value points to the next character in the string or to the null character at the end of the string, if the function is successful.

### Comments

The AnsiNext function can be used to move through strings where each character is a single byte, or through strings where each character is two or more bytes (such as strings that contain characters from a Japanese character set).

### Example

The following example uses the AnsiNext function to step through the characters in a filename:

```
/* Find the last backslash. */

for (lpszFile = lpszTemp; *lpszTemp != '\0';
    lpszTemp = AnsiNext(lpszTemp)) {

    if (*lpszTemp == '\\')
        lpszFile = AnsiNext(lpszTemp);
}
```

### See Also

[AnsiPrev](#)

## AnsiPrev (2.x)

```
LPSTR AnsiPrev(lpchStart, lpchCurrentChar)
LPCSTR lpchStart;          /* address of first character */
LPCSTR lpchCurrentChar;   /* address of current character */
/
```

The AnsiPrev function moves to the previous character in a string.

| Parameter       | Description                                        |
|-----------------|----------------------------------------------------|
| lpchStart       | Points to the beginning of the string.             |
| lpchCurrentChar | Points to a character in a null-terminated string. |

### Returns

The return value points to the previous character in the string, or to the first character in the string if the lpchCurrentChar parameter is equal to the lpchStart parameter.

### Comments

The AnsiPrev function can be used to move through strings where each character is a single byte, or through strings where each character is two or more bytes (such as strings that contain characters from a Japanese character set).

This function can be very slow, because the string must be scanned from the beginning to determine the previous character. Wherever possible, the [AnsiNext](#) function should be used instead of this function.

### Example

The following example uses the [AnsiNext](#) and AnsiPrev functions to change every occurrence of the characters '&' in a string to a single newline character:

```
/* Find ampersands. */
for (lpsz = lpszTest; *lpsz != '\0'; lpsz = AnsiNext(lpsz)) {
    /* Check the previous character. */
    if (*lpsz == '&' &&
        *(lpsz2 = AnsiPrev(lpszTest, lpsz)) == '\\') {
        lstrcpy(lpsz2, lpsz);
        *lpsz2 = '\n';
    }
}
```

See Also  
[AnsiNext](#)

## AnsiUpper (2.x)

```
LPSTR AnsiUpper(lpszString)
LPSTR lpszString;          /* address of string, or specific character */
```

The AnsiUpper function converts the given character string to uppercase.

| Parameter  | Description                                                                                                                                                                                                     |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszString | Points to a null-terminated string or specifies a single character. If the high-order word of this parameter is zero, the low-order byte of the low-order word must contain a single character to be converted. |

### Returns

The return value points to a converted character string if the function parameter is a character string. Otherwise, the return value is a 32-bit value that contains the converted character in the low-order byte of the low-order word.

### Comments

The language driver makes the conversion for the current language (the one selected by the user at setup or by using Control Panel). If no language driver is selected, Windows uses an internal function.

### Example

The following example uses the AnsiUpper function to convert two strings to uppercase for a non– case-sensitive comparison:

```
/*
 * Convert the target string to uppercase, and then
 * convert the subject string one character at a time.
 */

AnsiUpper(pszTarget);
while (*pszTarget != '\0') {
    if (*pszTarget != (char) (DWORD) AnsiUpper(
        MAKELP(0, *pszSubject)))
        return FALSE;
    pszTarget = AnsiNext(pszTarget);
    pszSubject = AnsiNext(pszSubject);
}
```

See Also

AnsiLower, AnsiUpperBuff

## AnsiUpperBuff (3.0)

```
UINT AnsiUpperBuff(lpszString, cbString)
LPSTR lpszString;      /* address of string to convert */
UINT cbString;        /* length of string*/
```

The AnsiUpperBuff function converts a character string in a buffer to uppercase.

| Parameter  | Description                                                                                                                          |
|------------|--------------------------------------------------------------------------------------------------------------------------------------|
| lpszString | Points to a buffer containing one or more characters.                                                                                |
| cbString   | Specifies the number of bytes in the buffer identified by the lpszString parameter. If cbString is zero, the length is 64K (65,536). |

### Returns

The return value specifies the length of the converted string if the function is successful.

### Comments

The language driver makes the conversion for the current language (the one selected by the user at setup or by using Control Panel). If no language driver is selected, Windows uses an internal function.

### Example

The following example uses the AnsiUpperBuff function to convert two strings to lowercase for a non-case-sensitive comparison:

```
/*
 * Convert both the subject and target strings to uppercase before
 * comparing.
 */
```

```
AnsiUpperBuff(pszSubject, (UINT) strlen(pszSubject));
AnsiUpperBuff(pszTarget, (UINT) strlen(pszTarget));
```

```
while (*pszTarget != '\0') {
    if (*pszTarget != *pszSubject)
        return FALSE;
    pszTarget = AnsiNext(pszTarget);
    pszSubject = AnsiNext(pszSubject);
}
```

### See Also

[AnsiLower](#), [AnsiUpper](#)

## AnyPopup (2.x)

BOOL AnyPopup(void)

The AnyPopup function indicates whether an unowned, visible, top-level pop-up, or overlapped window exists on the screen. The function searches the entire Windows screen, not just the caller's client area.

### Returns

The return value is nonzero if a pop-up window exists, even if the pop-up window is completely covered by other windows. The return value is zero if no pop-up window exists.

### Comments

AnyPopup is a Windows 1.x function and remains for compatibility reasons. It is generally not useful.

This function does not detect unowned pop-up windows or windows that do not have the WS\_VISIBLE style bit set.

### See Also

GetLastActivePopup, ShowOwnedPopups



## AppendMenu (3.0)

```
BOOL AppendMenu(hmenu, fuFlags, idNewItem, lpNewItem)
HMENU hmenu;          /* handle of menu */
UINT fuFlags;         /* menu-item
flags */
/
UINT idNewItem;
* menu-item identifier
/
LPCSTR lpNewItem;
* specifies menu-item content
/
```

The AppendMenu function appends a new item to the end of a menu. The application can specify the state of the menu item by setting values in the fuFlags parameter.

| Parameter           | Description                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hmenu               | Identifies the menu to be changed.                                                                                                                                                                                                                                                                                                                                                   |
| fuFlags             | Specifies information about the state of the new menu item when it is added to the menu. This parameter consists of one or more of the values listed in the following Comments section.                                                                                                                                                                                              |
| idNewItem           | Specifies either the command identifier of the new menu item or, if the fuFlags parameter is set to <u>MF_POPUP</u> , the menu handle of the pop-up menu.                                                                                                                                                                                                                            |
| lpNewItem           | Specifies the content of the new menu item. The interpretation of the lpNewItem parameter depends on the value of the fuFlags parameter.                                                                                                                                                                                                                                             |
| Value               | Menu-item content                                                                                                                                                                                                                                                                                                                                                                    |
| <u>MF_STRING</u>    | Contains a long pointer to a null-terminated string.                                                                                                                                                                                                                                                                                                                                 |
| <u>MF_BITMAP</u>    | Contains a bitmap handle in its low-order word.                                                                                                                                                                                                                                                                                                                                      |
| <u>MF_OWNERDRAW</u> | Contains an application-supplied 32-bit value that the application can use to maintain additional data associated with the menu item. An application can find this value in the itemData member of the structure pointed to by the lParam parameter of the <u>WM_MEASUREITEM</u> and <u>WM_DRAWITEM</u> messages that are sent when the menu item is changed or initially displayed. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

Whenever a menu changes (whether or not the menu is in a window that is displayed), the application should call the DrawMenuBar function.

Each of the following groups lists flags that are mutually exclusive and cannot be used together:

- MF\_DISABLED, MF\_ENABLED, and MF\_GRAYED
- MF\_BITMAP, MF\_STRING, and MF\_OWNERDRAW
- MF\_MENUBARBREAK and MF\_MENUBREAK
- MF\_CHECKED and MF\_UNCHECKED

Following are the flags that can be set in the fuFlags parameter:

| Value              | Meaning                                                                                                                                                                                                                    |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>MF_BITMAP</u>   | Uses a bitmap as the item. The low-order word of the lpNewItem parameter contains the handle of the bitmap.                                                                                                                |
| <u>MF_CHECKED</u>  | Places a check mark next to the item. If the application has supplied check mark bitmaps (see the <u>SetMenuItemBitmaps</u> function), setting this flag displays the "check mark on" <u>bitmap</u> next to the menu item. |
| <u>MF_DISABLED</u> | Disables the menu item so that it cannot be selected, but does not gray it.                                                                                                                                                |
| <u>MF_ENABLED</u>  | Enables the menu item so that it can be selected, and restores it from its grayed state.                                                                                                                                   |
| <u>MF_GRAYED</u>   | Disables the menu item so that it cannot be selected, and grays it.                                                                                                                                                        |

|                        |                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>MF_MENUBARBREAK</u> | Same as <u>MF_MENUBREAK</u> except that, for pop-up menus, separates the new column from the old column with a vertical line.                                                                                                                                                                                                                                                                         |
| <u>MF_MENUBREAK</u>    | Places the item on a new line for static menu-bar items. For pop-up menus, places the item in a new column, with no dividing line between the columns.                                                                                                                                                                                                                                                |
| <u>MF_OWNERDRAW</u>    | Specifies that the item is an owner-drawn item. The window that owns the menu receives a <u>WM_MEASUREITEM</u> message when the menu is displayed for the first time to retrieve the height and width of the menu item. The <u>WM_DRAWITEM</u> message is then sent whenever the owner window must update the visual appearance of the menu item. This option is not valid for a top-level menu item. |
| <u>MF_POPUP</u>        | Specifies that the menu item has a pop-up menu associated with it. The <u>idNewItem</u> parameter specifies a handle to a pop-up menu to be associated with the item. This is used for adding either a top-level pop-up menu or adding a hierarchical pop-up menu to a pop-up menu item.                                                                                                              |
| <u>MF_SEPARATOR</u>    | Draws a horizontal dividing line. Can be used only in a pop-up menu. This line cannot be grayed, disabled, or highlighted. The <u>lpNewItem</u> and <u>idNewItem</u> parameters are ignored.                                                                                                                                                                                                          |
| <u>MF_STRING</u>       | Specifies that the menu item is a character string; the <u>lpNewItem</u> parameter points to the string for the menu item.                                                                                                                                                                                                                                                                            |
| <u>MF_UNCHECKED</u>    | Does not place a check mark next to the item (default). If the application has supplied check mark bitmaps (see <u>SetMenuItemBitmaps</u> ), setting this flag displays the "check mark off" bitmap next to the menu item.                                                                                                                                                                            |

#### Example

The following example uses the AppendMenu function to append three items to a floating pop-up menu:

```

POINT ptCurrent;
HMENU hmenu;

ptCurrent = MAKEPOINT(lParam);
hmenu = CreatePopupMenu();
AppendMenu(hmenu, MF_ENABLED, IDM_ELLIPSE, "Ellipse");
AppendMenu(hmenu, MF_ENABLED, IDM_SQUARE, "Square");
AppendMenu(hmenu, MF_ENABLED, IDM_TRIANGLE, "Triangle");
ClientToScreen(hwnd, &ptCurrent);
TrackPopupMenu(hmenu, TPM_LEFTALIGN, ptCurrent.x,
ptCurrent.y, 0, hwnd, NULL);

```

#### See Also

CreateMenu, DeleteMenu, DrawMenuBar, InsertMenu, RemoveMenu, SetMenuItemBitmaps

MF\_BITMAP 0x0004

Uses a bitmap as the item. The low-order word of the lpNewItem parameter contains the handle of the bitmap.

MF\_BITMAP 0x0004

MF\_CHECKED 0x0008

Places a check mark next to the item. If the application has supplied check mark bitmaps (see the SetMenuItemBitmaps function), setting this flag displays the "check mark on" bitmap next to the menu item.

MF\_CHECKED 0x0008

MF\_DISABLED 0x0002

Disables the menu item so that it cannot be selected, but does not gray it.

MF\_DISABLED 0x0002



MF\_ENABLED 0x0000

Enables the menu item so that it can be selected, and restores it from its grayed state.

MF\_ENABLED 0x0000

MF\_GRAYED 0x0001

Disables the menu item so that it cannot be selected, and grays it.

MF\_GRAYED 0x0001

MF\_MENUBARBREAK 0x0020

Same as MF\_MENUBREAK except that, for pop-up menus, separates the new column from the old column with a vertical line.

MF\_MENUBARBREAK 0x0020

MF\_MENUBREAK 0x0040

Places the item on a new line for static menu-bar items. For pop-up menus, places the item in a new column, with no dividing line between the columns.

MF\_MENUBREAK 0x0040



MF\_OWNERDRAW 0x0100

Specifies that the item is an owner-drawn item. The window that owns the menu receives a WM\_MEASUREITEM message when the menu is displayed for the first time to retrieve the height and width of the menu item. The WM\_DRAWITEM message is then sent whenever the owner window must update the visual appearance of the menu item. This option is not valid for a top-level menu item.

MF\_OWNERDRAW 0x0100

MF\_POPUP 0x0010

Specifies that the menu item has a pop-up menu associated with it. The `idNewItem` parameter specifies a handle to a pop-up menu to be associated with the item. This is used for adding either a top-level pop-up menu or adding a hierarchical pop-up menu to a pop-up menu item.

MF\_POPUP 0x0010

MF\_SEPARATOR 0x0800

Draws a horizontal dividing line. Can be used only in a pop-up menu. This line cannot be grayed, disabled, or highlighted. The lpNewItem and idNewItem parameters are ignored.

MF\_SEPARATOR 0x0800

MF\_STRING 0x0000

Specifies that the menu item is a character string; the lpNewItem parameter points to the string for the menu item.

MF\_STRING 0x0000



MF\_UNCHECKED 0x0000

Does not place a check mark next to the item (default). If the application has supplied check mark bitmaps (see SetMenuItemBitmaps), setting this flag displays the "check mark off" bitmap next to the menu item.

MF\_UNCHECKED 0x0000

## ArrangeIconicWindows (3.0)

```
UINT ArrangeIconicWindows(hwnd)
HWND hwnd;    /* handle of parent window */
```

The ArrangeIconicWindows function arranges all the minimized (iconic) child windows of a parent window.

| Parameter | Description                   |
|-----------|-------------------------------|
| hwnd      | Identifies the parent window. |

### Returns

The return value is the height of one row of icons if the function is successful. Otherwise, it is zero.

### Comments

An application that maintains its own minimized child windows can call ArrangeIconicWindows to arrange icons in a client window. This function also arranges icons on the desktop window, which covers the entire screen. The [GetDesktopWindow](#) function retrieves the window handle of the desktop window.

An application sends the [WM\\_MDIICONARRANGE](#) message to the MDI client window to prompt the client window to arrange its minimized MDI child windows.

### See Also

[GetDesktopWindow](#)

## BeginDeferWindowPos (3.0)

```
HDWP BeginDeferWindowPos(cWindows)
int cWindows; /* number of windows */
```

The BeginDeferWindowPos function returns a handle of an internal structure. The DeferWindowPos function fills this structure with information about the target position for a window that is about to be moved. The EndDeferWindowPos function accepts a handle of this structure and instantaneously repositions the windows by using the information stored in the structure.

| Parameter | Description                                                                                                                                                                            |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cWindows  | Specifies the initial number of windows for which to store position information in the structure. The <u>DeferWindowPos</u> function increases the size of the structure if necessary. |

### Returns

The return value identifies the internal structure if the function is successful. Otherwise, it is NULL.

### Comments

If Windows must increase the size of the internal structure beyond the initial size specified by the cWindows parameter but cannot allocate enough memory to do so, Windows fails the entire begin/defer/end window-positioning sequence. By specifying the maximum size needed, an application can detect and handle failure early in the process.

### See Also

DeferWindowPos, EndDeferWindowPos

## BeginPaint (2.x)

```
HDC BeginPaint(hwnd, lpps)
HWND hwnd; /* handle of window to paint */
PAINTSTRUCT FAR* lpps;
/* address of structure with paint information */
/
```

The `BeginPaint` function prepares the specified window for painting and fills a `PAINTSTRUCT` structure with information about the painting.

| Parameter         | Description                                                                                         |
|-------------------|-----------------------------------------------------------------------------------------------------|
| <code>hwnd</code> | Identifies the window to be repainted.                                                              |
| <code>lpps</code> | Points to the <u><code>PAINTSTRUCT</code></u> structure that will receive the painting information. |

### Returns

The return value is the handle of the device context for the given window if the function is successful.

### Comments

The `BeginPaint` function automatically sets the clipping region of the device context to exclude any area outside the update region. The update region is set by the `InvalidateRect` or `InvalidateRgn` function and by the system after sizing, moving, creating, scrolling, or any other operation that affects the client area. If the update region is marked for erasing, `BeginPaint` sends a `WM_ERASEBKGND` message to the window.

An application should not call `BeginPaint` except in response to a `WM_PAINT` message. Each call to the `BeginPaint` function must have a corresponding call to the `EndPaint` function.

If the caret is in the area to be painted, `BeginPaint` automatically hides the caret to prevent it from being erased.

If the window's class has a background brush, `BeginPaint` will use that brush to erase the background of the update region before returning.

### Example

The following example calls an application-defined function to paint a bar graph in a window's client area during the `WM_PAINT` message:

```
PAINTSTRUCT ps;

case WM_PAINT:
    BeginPaint(hwnd, &ps);
    .
    .
    .
    EndPaint(hwnd, &ps);
    break;
```

### See Also

`EndPaint`, `InvalidateRect`, `InvalidateRgn`, `ValidateRect`, `ValidateRgn`, `PAINTSTRUCT`, `RECT`, `WM_PAINT`, `WM_ERASEBKGND`

## BringWindowToTop (2.x)

```
BOOL BringWindowToTop(hwnd)
HWND hwnd;    /* handle of window */
```

The BringWindowToTop function brings the given pop-up or child window (including an MDI child window) to the top of a stack of overlapping windows. In addition, it activates pop-up, top-level, and MDI child windows. The BringWindowToTop function should be used to uncover any window that is partially or completely obscured by any overlapping windows.

| Parameter | Description                                                |
|-----------|------------------------------------------------------------|
| hwnd      | Identifies the pop-up or child window to bring to the top. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

Calling this function is similar to calling the SetWindowPos function to change a window's position in the Z-order. The BringWindowToTop function does not make a window a top-level window.

### See Also

SetWindowPos

## BuildCommDCB (2.x)

```
int BuildCommDCB(lpszDef, lpdcB)
LPCSTR lpszDef;      /* address of device-control string */
DCB FAR* lpdcB;     /* address of
device-control block
/
```

The BuildCommDCB function translates a device-definition string into appropriate serial device control block (DCB) codes.

| Parameter | Description                                                                                                                                                         |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszDef   | Points to a null-terminated string that specifies device-control information. The string must have the same form as the parameters used in the MS-DOS mode command. |
| lpdcB     | Points to a <u>DCB</u> structure that will receive the translated string. The structure defines the control settings for the serial-communications device.          |

### Returns

The return value is zero if the function is successful. Otherwise, it is -1.

### Example

The following example uses the BuildCommDCB and SetCommState functions to set up COM1 to operate at 9600 baud, with no parity, 8 data bits, and 1 stop bit:

```
idComDev = OpenComm("COM1", 1024, 128);
if (idComDev < 0) {
    ShowError(idComDev, "OpenComm");
    return 0;
}

err = BuildCommDCB("COM1:9600,n,8,1", &dcB);
if (err < 0) {
    ShowError(err, "BuildCommDCB");
    return 0;
}

err = SetCommState(&dcB);
if (err < 0) {
    ShowError(err, "SetCommState");
    return 0;
}
```

### Comments

The BuildCommDCB function only fills the buffer. To apply the settings to a port, an application should use the SetCommState function.

By default, BuildCommDCB specifies XON/XOFF and hardware flow control as disabled. To enable flow control, an application should set the appropriate members in the DCB structure.

### See Also

SetCommState, DCB

## CallMsgFilter (2.x)

```
BOOL CallMsgFilter(lpmsg, nCode)
MSG FAR* lpmsg;      /* address of structure with message data */
int nCode;           /* processing */
code
/
```

The CallMsgFilter function passes the given message and code to the current message-filter function. The message-filter function is an application-specified function that examines and modifies all messages. An application specifies the function by using the SetWindowsHook function.

| Parameter | Description                                                                           |
|-----------|---------------------------------------------------------------------------------------|
| lpmsg     | Points to an <u>MSG</u> structure that contains the message to be filtered.           |
| nCode     | Specifies a code used by the filter function to determine how to process the message. |

### Returns

The return value specifies the state of message processing. It is zero if the message should be processed or nonzero if the message should not be processed further.

### Comments

The CallMsgFilter function is usually called by Windows to let applications examine and control the flow of messages during internal processing in menus and scroll bars or when moving or sizing a window.

Values given for the nCode parameter must not conflict with any of the MSGF\_ and HC\_ values passed by Windows to the message-filter function.

### See Also

SetWindowsHook, MSG



### CallNextHookEx (3.1)

```
LRESULT CallNextHookEx(hHook, nCode, wParam, lParam)
HHOOK hHook;          /* handle of hook function */
int nCode;            /* hook code */
WPARAM wParam;
/* first message parameter
/
LPARAM lParam;
/* second message parameter
/
```

The CallNextHookEx function passes the hook information to the next hook function in the hook chain.

| Parameter | Description                                                                                                                                         |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| hHook     | Identifies the current hook function.                                                                                                               |
| nCode     | Specifies the hook code to pass to the next hook function. A hook function uses this code to determine how to process the message sent to the hook. |
| wParam    | Specifies 16 bits of additional message-dependent information.                                                                                      |
| lParam    | Specifies 32 bits of additional message-dependent information.                                                                                      |

#### Returns

The return value specifies the result of the message processing and depends on the value of the nCode parameter.

#### Comments

Calling the CallNextHookEx function is optional. An application can call this function either before or after completing any processing in its own hook function. If an application does not call CallNextHookEx, Windows will not call the hook functions that were installed before the application's hook function was installed.

#### See Also

[SetWindowsHookEx](#), [UnhookWindowsHookEx](#)

## CallWindowProc (2.x)

```
LRESULT CallWindowProc(wndprcPrev, hwnd, uMsg, wParam, lParam)
WNDPROC wndprcPrev; /* instance address of previous procedure */
HWND hwnd; /* handle of window */
/
UINT uMsg;
* message
/
WPARAM wParam;
* first message parameter
/
LPARAM lParam;
* second message parameter
/
```

The CallWindowProc function passes message information to the specified window procedure.

| Parameter  | Description                                                                |
|------------|----------------------------------------------------------------------------|
| wndprcPrev | Specifies the procedure-instance address of the previous window procedure. |
| hwnd       | Identifies the window that will receive the message.                       |
| uMsg       | Specifies the message.                                                     |
| wParam     | Specifies 16 bits of additional message-dependent information.             |
| lParam     | Specifies 32 bits of additional message-dependent information.             |

### Returns

The return value specifies the result of the message processing and depends on the message sent.

### Comments

The CallWindowProc function is used for window subclassing. Normally, all windows with the same class share the same window procedure. A subclass is a window or set of windows belonging to the same window class whose messages are intercepted and processed by another window procedure (or procedures) before being passed to the window procedure of that class.

The SetWindowLong function creates the subclass by changing the window procedure associated with a particular window, causing Windows to call the new window procedure instead of the previous one. Any messages not processed by the new window procedure must be passed to the previous window procedure by calling CallWindowProc. This allows you to create a chain of window procedures.

### See Also

[SetWindowLong](#)

## ChangeClipboardChain (2.x)

```
BOOL ChangeClipboardChain(HWND hwnd, HWND hwndNext)
/* handle of window to remove */
/* handle of next
window
/
```

The ChangeClipboardChain function removes the window identified by the hwnd parameter from the chain of clipboard viewers and makes the window identified by the hwndNext parameter the descendant of the hwnd parameter's ancestor in the chain.

| Parameter | Description                                                                                                                                                                                                             |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window that is to be removed from the chain. The handle must have been passed to the <u>SetClipboardViewer</u> function.                                                                                 |
| hwndNext  | Identifies the window that follows hwnd in the clipboard-viewer chain (this is the handle returned by the SetClipboardViewer function, unless the sequence was changed in response to a <u>WM_CHANGECHAIN</u> message). |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### See Also

SetClipboardViewer, WM\_CHANGECHAIN

## ChangeMenu (2.x)

The Microsoft Windows 3.1 Software Development Kit (SDK) has replaced this function with five specialized functions, listed as follows:

| Function          | Description                                                         |
|-------------------|---------------------------------------------------------------------|
| <u>AppendMenu</u> | Appends a menu item to the end of a menu.                           |
| <u>DeleteMenu</u> | Deletes a menu item from a menu, destroying the menu item.          |
| <u>InsertMenu</u> | Inserts a menu item into a menu.                                    |
| <u>ModifyMenu</u> | Modifies a menu item in a menu.                                     |
| <u>RemoveMenu</u> | Removes a menu item from a menu but does not destroy the menu item. |

Applications written for Windows versions earlier than 3.0 may continue to call ChangeMenu as previously documented. Applications written for Windows 3.0 and 3.1 should call the new functions.

### Example

The following example shows a call to ChangeMenu and how it would be rewritten to call AppendMenu:

```
ChangeMenu(hMenu, /* handle of menu */
           0, /* position parameter not used */
           "&White", /* menu-item string */
           IDM_PATTERN1, /* menu-item identifier */
           MF_APPEND | MF_STRING | MF_CHECKED); /* flags */

AppendMenu(hMenu, /* handle of menu */
           MF_STRING | MF_CHECKED, /* flags */
           IDM_PATTERN1, /* menu-item identifier */
           "&White"); /* menu-item string*/
```

### See Also

AppendMenu, DeleteMenu, InsertMenu, ModifyMenu, RemoveMenu

## CheckDlgButton (2.x)

```
void CheckDlgButton(HWND hDlg, int idButton, int uCheck)
/* handle of dialog box */
/* button-control identifier */
int idButton;
UINT uCheck;
* check state
/
```

The CheckDlgButton function selects (places a check mark next to) or clears (removes a check mark from) a button control, or it changes the state of a three-state button.

| Parameter | Description                                                                                                                                                                                                                                                                                           |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hDlg      | Identifies the dialog box that contains the button.                                                                                                                                                                                                                                                   |
| idButton  | Identifies the button to be modified.                                                                                                                                                                                                                                                                 |
| uCheck    | Specifies the check state of the button. If this parameter is nonzero, CheckDlgButton selects the button; if the parameter is zero, the function clears the button. For a three-state check box, if uCheck is 2, the button is grayed; if uCheck is 1, it is selected; if uCheck is 0, it is cleared. |

### Returns

This function does not return a value.

### Comments

The CheckDlgButton function sends a BM\_SETCHECK message to the specified button control in the given dialog box.

### See Also

CheckRadioButton, IsDlgButtonChecked, BM\_GETCHECK, BM\_SETCHECK

## CheckMenuItem (2.x)

```
BOOL CheckMenuItem(hmenu, idCheckItem, uCheck)
HMENU hmenu;          /* handle of menu */
UINT idCheckItem;     /* menu-item identifier */
UINT uCheck;          /* check state and position */
```

The CheckMenuItem function selects (places a check mark next to) or clears (removes a check mark from) a specified menu item in the given pop-up menu.

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                 |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hmenu       | Identifies the menu.                                                                                                                                                                                                                                                                                                                                        |
| idCheckItem | Identifies the menu item to be selected or cleared.                                                                                                                                                                                                                                                                                                         |
| uCheck      | Specifies how to determine the position of the menu item ( <u>MF_BYCOMMAND</u> or <u>MF_BYPOSITION</u> ) and whether the item should be selected or cleared ( <u>MF_CHECKED</u> or <u>MF_UNCHECKED</u> ). This parameter can be a combination of these values, which can be combined by using the bitwise OR operator. The values are described as follows: |

| Value                | Meaning                                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------------------------------|
| <u>MF_BYCOMMAND</u>  | Specifies that the idCheckItem parameter gives the menu-item identifier ( <u>MF_BYCOMMAND</u> is the default).     |
| <u>MF_BYPOSITION</u> | Specifies that the idCheckItem parameter gives the position of the menu item (the first item is at position zero). |
| <u>MF_CHECKED</u>    | Selects the item (adds check mark).                                                                                |
| <u>MF_UNCHECKED</u>  | Clears the item (removes check mark).                                                                              |

### Returns

The return value specifies the previous state of the item--MF\_CHECKED or MF\_UNCHECKED--if the function is successful. The return value is -1 if the menu item does not exist.

### Comments

The idCheckItem parameter may identify a pop-up menu item as well as a menu item. No special steps are required to select a pop-up menu item.

Top-level menu items cannot have a check.

A pop-up menu item should be selected by position since it does not have a menu-item identifier associated with it.

### See Also

GetMenuState, SetMenuItemBitmaps

## CheckRadioButton (2.x)

```
void CheckRadioButton(HWND hwndDlg, int idFirstButton, int idLastButton, int idCheckButton)
/* handle of dialog box */
int idFirstButton; /* identifier of
first radio button in group
/
int idLastButton;
* identifier of last radio button in group
/
int idCheckButton;
* identifier of radio button to select
/
```

The CheckRadioButton function selects (adds a check mark to) a given radio button in a group and clears (removes a check mark from) all other radio buttons in the group.

| Parameter     | Description                                                      |
|---------------|------------------------------------------------------------------|
| hwndDlg       | Identifies the dialog box that contains the radio button.        |
| idFirstButton | Specifies the identifier of the first radio button in the group. |
| idLastButton  | Specifies the identifier of the last radio button in the group.  |
| idCheckButton | Specifies the identifier of the radio button to select.          |

### Returns

This function does not return a value.

### Comments

The CheckRadioButton function sends a BM\_SETCHECK message to the specified radio button control in the given dialog box.

### See Also

CheckDlgButton, IsDlgButtonChecked, BM\_GETCHECK, BM\_SETCHECK

## ChildWindowFromPoint (2.x)

```
HWND ChildWindowFromPoint(HWND hwndParent, POINT pt);
/* handle of parent window */
/* structure with point coordinates */
```

The ChildWindowFromPoint function determines which, if any, of the child windows belonging to the given parent window contains the specified point.

| Parameter  | Description                                                                                        |
|------------|----------------------------------------------------------------------------------------------------|
| hwndParent | Identifies the parent window.                                                                      |
| pt         | Specifies a <u>POINT</u> structure that defines the client coordinates of the point to be checked. |

### Returns

The return value is the handle of the child window (hidden, disabled, or transparent) that contains the point, if the function is successful. If the given point lies outside the parent window, the return value is NULL. If the point is within the parent window but is not contained within any child window, the return value is the handle of the parent window.

### Comments

More than one window may contain the given point, but Windows returns the handle only of the first window encountered that contains the point.

### See Also

WindowFromPoint



## ClearCommBreak (2.x)

```
int ClearCommBreak(idComDev)
int idComDev;      /* device to be restored */
```

The ClearCommBreak function restores character transmission and places the communications device in a nonbreak state.

| Parameter | Description                                                                                           |
|-----------|-------------------------------------------------------------------------------------------------------|
| idComDev  | Identifies the communications device to be restored. The <u>OpenComm</u> function returns this value. |

### Returns

The return value is zero if the function is successful, or -1 if the idComDev parameter does not identify a valid device.

### Comments

This function clears the communications-device break state set by the SetCommBreak function.

### See Also

OpenComm, SetCommBreak

## ClientToScreen (2.x)

```
void ClientToScreen(hwnd, lppt)
HWND hwnd;          /* window handle for source coordinates */
POINT FAR* lppt;    /*
address of structure with coordinates
/
```

The ClientToScreen function converts the client coordinates of a given point on the screen to screen coordinates.

| Parameter | Description                                                                              |
|-----------|------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window whose client area is used for the conversion.                      |
| lppt      | Points to a <u>POINT</u> structure that contains the client coordinates to be converted. |

### Returns

This function does not return a value.

### Comments

The ClientToScreen function replaces the coordinates in the POINT structure with the screen coordinates. The screen coordinates are relative to the upper-left corner of the screen.

### Example

The following example uses the LOWORD and HIWORD macros and the ClientToScreen function to convert the mouse position to screen coordinates:

```
POINT pt;

pt.x = LOWORD(lParam);
pt.y = HIWORD(lParam);
ClientToScreen(hwnd, &pt);
```

### See Also

MapWindowPoints, ScreenToClient

## ClipCursor (2.x)

```
void ClipCursor(lprc)
const RECT FAR* lprc;          /* address of structure with rectangle */
```

The ClipCursor function confines the cursor to a rectangle on the screen. If a subsequent cursor position (set by the [SetCursorPos](#) function or by the mouse) lies outside the rectangle, Windows automatically adjusts the position to keep the cursor inside.

| Parameter | Description                                                                                                                                                                                                                         |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lprc      | Points to a <a href="#">RECT</a> structure that contains the screen coordinates of the upper-left and lower-right corners of the confining rectangle. If this parameter is NULL, the cursor is free to move anywhere on the screen. |

### Returns

This function does not return a value.

### Comments

The cursor is a shared resource. An application that has confined the cursor to a given rectangle must free it before relinquishing control to another application.

### See Also

[GetClipCursor](#), [GetCursorPos](#), [SetCursorPos](#), [RECT](#)

## CloseClipboard (2.x)

BOOL CloseClipboard(void)

The CloseClipboard function closes the clipboard.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The CloseClipboard function should be called when a window has finished examining or changing the clipboard. This lets other applications access the clipboard.

### See Also

GetOpenClipboardWindow, OpenClipboard

## CloseComm (2.x)

```
int CloseComm(idComDev)
int idComDev;      /* device to close */
```

The CloseComm function closes the specified communications device and frees any memory allocated for the device's transmission and receiving queues. All characters in the output queue are sent before the communications device is closed.

| Parameter | Description                                                                         |
|-----------|-------------------------------------------------------------------------------------|
| idComDev  | Specifies the device to be closed. The <u>OpenComm</u> function returns this value. |

### Returns

The return value is zero if the function is successful. Otherwise, it is less than zero.

### See Also

OpenComm

## CloseWindow (2.x)

```
void CloseWindow(hwnd)
HWND hwnd;    /* handle of window to minimize */
```

The CloseWindow function minimizes (but does not destroy) the given window. To destroy a window, an application must use the DestroyWindow function.

| Parameter | Description                            |
|-----------|----------------------------------------|
| hwnd      | Identifies the window to be minimized. |

### Returns

This function does not return a value.

### Comments

This function has no effect if the hwnd parameter identifies a pop-up or child window.

### See Also

DestroyWindow, IsIconic, OpenIcon

### CloseDriver (3.1)

```
LRESULT CloseDriver(hdrvr, lParam1, lParam2)
HDRVR hdrvr;          /* handle of installable driver */
LPARAM lParam1;      /* driver-specific data */
/* driver-specific data */
LPARAM lParam2;
/* driver-specific data */
```

The CloseDriver function closes an installable driver.

| Parameter | Description                                                                                                                                  |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------|
| hdrvr     | Identifies the installable driver to be closed. This parameter must have been obtained by a previous call to the <u>OpenDriver</u> function. |
| lParam1   | Specifies driver-specific data.                                                                                                              |
| lParam2   | Specifies driver-specific data.                                                                                                              |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

When an application calls CloseDriver and the driver identified by hdrvr is the last instance of the driver, Windows calls the DriverProc function three times. On the first call, Windows sets the third DriverProc parameter, wMessage, to DRV\_CLOSE; on the second call, Windows sets wMessage to DRV\_DISABLE; and on the third call, Windows sets wMessage to DRV\_FREE. When the driver identified by hdrvr is not the last instance of the driver, only DRV\_CLOSE is sent. The values specified in the lParam1 and lParam2 parameters are passed to the lParam1 and lParam2 parameters of the DriverProc function.

#### See Also

DriverProc, OpenDriver

### CopyCursor (3.1)

```
HCURSOR CopyCursor(hinst, hcur)
HINSTANCE hinst;          /* handle of application instance */
HCURSOR hcur;            /* handle of
cursor to copy
/
```

The CopyCursor function copies a cursor.

| Parameter | Description                                                      |
|-----------|------------------------------------------------------------------|
| hinst     | Identifies the instance of the module that will copy the cursor. |
| hcur      | Identifies the cursor to be copied.                              |

#### Returns

The return value is the handle of the duplicate cursor if the function is successful. Otherwise, it is NULL.

#### Comments

When it no longer requires a cursor, an application must destroy the cursor, using the [DestroyCursor](#) function.

The CopyCursor function allows an application or dynamic-link library to accept a cursor from another module. Because all resources are owned by the module in which they originate, a resource cannot be shared after the module is freed. CopyCursor allows an application to create a copy that the application then owns.

#### See Also

[CopyIcon](#), [DestroyCursor](#), [GetCursor](#), [SetCursor](#), [ShowCursor](#)



### CopyIcon (3.1)

```
HICON CopyIcon(hinst, hicon)
HINSTANCE hinst;          /* handle of application instance */
HICON hicon;              /* handle of icon */
to copy
/
```

The CopyIcon function copies an icon.

| Parameter | Description                                                    |
|-----------|----------------------------------------------------------------|
| hinst     | Identifies the instance of the module that will copy the icon. |
| hicon     | Identifies the icon to be copied.                              |

#### Returns

The return value is the handle of the duplicate icon if the function is successful. Otherwise, it is NULL.

#### Comments

When it no longer requires an icon, an application should destroy the icon, using the [DestroyIcon](#) function.

The CopyIcon function allows an application or dynamic-link library to accept an icon from another module. Because all resources are owned by the module in which they originate, a resource cannot be shared after the module is freed. CopyIcon allows an application to create a copy that the application then owns.

#### See Also

[CopyCursor](#), [DestroyIcon](#), [DrawIcon](#)

## CopyRect (2.x)

```
void CopyRect(lprcDst, lprcSrc)
RECT FAR* lprcDst;          /* address of struct. for destination rect. */
const RECT FAR* lprcSrc;    /* address of struct. with source rect. */
/
```

The CopyRect function copies the dimensions of one rectangle to another.

| Parameter | Description                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------|
| lprcDst   | Points to the <u>RECT</u> structure that will receive the dimensions of the source rectangle. |
| lprcSrc   | Points to the <u>RECT</u> structure whose dimensions are to be copied.                        |

### Returns

This function does not return a value.

### See Also

SetRect, RECT

## CountClipboardFormats (2.x)

int CountClipboardFormats(void)

The CountClipboardFormats function retrieves the number of different data formats currently in the clipboard.

### Returns

The return value specifies the number of different data formats in the clipboard, if the function is successful.

### See Also

[EnumClipboardFormats](#)

## CreateCaret (2.x)

```
void CreateCaret(HWND hwnd, HBITMAP hbmp, int nWidth, int nHeight)
/* handle of owner window */
/* handle of
bitmap for caret shape
/
int nWidth;
* caret width
/
int nHeight;
* caret height
/
```

The CreateCaret function creates a new shape for the system caret and assigns ownership of the caret to the given window. The caret shape can be a line, block, or bitmap.

| Parameter | Description                                                                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window that owns the new caret.                                                                                                |
| hbmp      | Identifies the bitmap that defines the caret shape. If this parameter is NULL, the caret is solid; if the parameter is 1, the caret is gray.  |
| nWidth    | Specifies the width of the caret in logical units. If this parameter is NULL, the width is set to the system-defined window-border width.     |
| nHeight   | Specifies the height of the caret, in logical units. If this parameter is NULL, the height is set to the system-defined window-border height. |

### Returns

This function does not return a value.

### Comments

If the hbmp parameter contains a bitmap handle, the nWidth and nHeight parameters are ignored; the bitmap defines its own width and height. (The bitmap handle must have been created by using the [CreateBitmap](#), [CreateDIBitmap](#), or [LoadBitmap](#) function.) If hbmp is NULL or 1, nWidth and nHeight give the caret's width and height, in logical units; the exact width and height (in pixels) depend on the window's mapping mode.

The CreateCaret function automatically destroys the previous caret shape, if any, regardless of which window owns the caret. Once created, the caret is initially hidden. To show the caret, use the [ShowCaret](#) function.

The system caret is a shared resource. A window should create a caret only when it has the input focus or is active. It should destroy the caret before losing the input focus or becoming inactive.

The system's window-border width or height can be retrieved by using the [GetSystemMetrics](#) function, specifying the [SM\\_CXBORDER](#) and [SM\\_CYBORDER](#) indices. Using the window-border width or height guarantees that the caret will be visible on a high-resolution screen.

### Example

The following example creates a caret, sets its initial position, and then displays the caret:

```
case WM_SETFOCUS:
    CreateCaret(hwndParent, NULL, CARET_WIDTH, CARET_HEIGHT);
    SetCaretPos(CARET_XPOS, CARET_YPOS);
    ShowCaret(hwndParent);
    break;
```

### See Also

[CreateBitmap](#), [CreateDIBitmap](#), [DestroyCaret](#), [GetSystemMetrics](#), [LoadBitmap](#), [ShowCaret](#)

### CreateCursor (3.0)

```
HCURSOR CreateCursor(hinst, xHotSpot, yHotSpot, nWidth, nHeight, lpvANDplane, lpvXORplane)
HINSTANCE hinst;          /* handle of application instance */
int xHotSpot;             /* horizontal position of hot spot */
int yHotSpot;            /* vertical position of hot spot */
int nWidth;               /* cursor width */
int nHeight;              /* cursor height */
const void FAR* lpvANDplane; /* address of AND mask array */
const void FAR* lpvXORplane; /* address of XOR mask array */
```

The CreateCursor function creates a cursor that has the specified width, height, and bit patterns.

| Parameter   | Description                                                                                                                                             |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst       | Identifies the instance of the module that will create the cursor.                                                                                      |
| xHotSpot    | Specifies the horizontal position of the cursor hot spot.                                                                                               |
| yHotSpot    | Specifies the vertical position of the cursor hot spot.                                                                                                 |
| nWidth      | Specifies the width, in pixels, of the cursor.                                                                                                          |
| nHeight     | Specifies the height, in pixels, of the cursor.                                                                                                         |
| lpvANDplane | Points to an array of bytes that contains the bit values for the AND mask of the cursor. These can be the bits of a device-dependent monochrome bitmap. |
| lpvXORplane | Points to an array of bytes that contains the bit values for the XOR mask of the cursor. These can be the bits of a device-dependent monochrome bitmap. |

#### Returns

The return value is the handle of the cursor if the function is successful. Otherwise, it is NULL.

#### Comments

The nWidth and nHeight parameters must specify a width and height supported by the current display driver, since the system cannot create cursors of other sizes. An application can determine the width and height supported by the display driver by calling the [GetSystemMetrics](#) function and specifying the [SM\\_CXCURSOR](#) or [SM\\_CYCURSOR](#) value.

Before terminating, an application must call the [DestroyCursor](#) function to free any system resources associated with the cursor.

#### See Also

[CreateIcon](#), [DestroyCursor](#), [GetSystemMetrics](#), [SetCursor](#)

## CreateDialog (2.x)

```
HWND CreateDialog(hinst, lpszDlgTemp, hwndOwner, dlgprc)
HINSTANCE hinst;          /* handle of application instance */
LPCSTR lpszDlgTemp;
/* address of dialog box template name
/
HWND hwndOwner;
/* handle of owner window
/
DLGPROC dlgprc;
/* instance address of dialog box procedure
/
```

The CreateDialog function creates a modeless dialog box from a dialog box template resource.

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst       | Identifies an instance of the module whose executable file contains the dialog box template.                                                                                                                                                                                                                                               |
| lpszDlgTemp | Points to a null-terminated string that names the dialog box template.                                                                                                                                                                                                                                                                     |
| hwndOwner   | Identifies the window that owns the dialog box.                                                                                                                                                                                                                                                                                            |
| dlgprc      | Specifies the procedure-instance address of the dialog box procedure. The address must be created by using the <u>MakeProcInstance</u> function, except when the function and dialog box procedure are used in a DLL. For more information about the dialog box procedure, see the description of the <u>DialogProc</u> callback function. |

### Returns

The return value is the handle of the dialog box that was created, if the function is successful. Otherwise, it is NULL.

### Comments

The CreateWindowEx function is called to create the dialog box. The dialog box procedure then receives a WM\_SETFONT message (if the DS\_SETFONT style was specified) and a WM\_INITDIALOG message, and then the dialog box is displayed.

The CreateDialog function returns immediately after creating the dialog box.

To make the dialog box appear in the owner window upon being created, use the WS\_VISIBLE style in the dialog box template.

Use the DestroyWindow function to destroy a dialog box created by the CreateDialog function.

A dialog box can contain up to 255 controls.

### Example

The following example creates a modeless dialog box:

```
HWND hwndDlgFindBox;
DLGPROC dlgprc = (DLGPROC) MakeProcInstance(FindDlgProc, hinst);

hwndDlgFindBox = CreateDialog(hinst, "dlgFindBox", hwndParent, dlgprc)
;
```

### See Also

CreateDialogIndirect, CreateDialogIndirectParam, CreateDialogParam, DestroyWindow, MakeProcInstance, WM\_INITDIALOG

## CreateDialogIndirect (2.x)

```
HWND CreateDialogIndirect(hinst, lpvDlgTmp, hwndOwner, dlgprc)
HINSTANCE hinst;          /* handle of application instance */
const void FAR* lpvDlgTmp;
/* address of dialog box template
/
HWND hwndOwner;
/* handle of owner window
/
DLGPROC dlgprc;
/* instance address of dialog box procedure
/
```

The CreateDialogIndirect function creates a modeless dialog box from a dialog box template in memory.

| Parameter | Description                                                                                                                                                                                                                                                                                                                   |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst     | Identifies the instance of the module that will create the dialog box.                                                                                                                                                                                                                                                        |
| lpvDlgTmp | Points to a global memory object that contains a dialog box template used to create the dialog box. This template is in the form of a <a href="#">DialogBoxHeader</a> structure. For more information about this structure, see the <a href="#">Dialog Box Resource</a> topic.                                                |
| hwndOwner | Identifies the window that owns the dialog box.                                                                                                                                                                                                                                                                               |
| dlgprc    | Specifies the procedure-instance address of the dialog box procedure. The address must be created by using the <a href="#">MakeProcInstance</a> function, except when the function and dialog box procedure are used in a DLL. For more information, see the description of the <a href="#">DialogProc</a> callback function. |

### Returns

The return value is the window handle of the dialog box if the function is successful. Otherwise, it is NULL.

### Comments

The [CreateWindowEx](#) function is called to create the dialog box. The dialog box procedure then receives a [WM\\_SETFONT](#) message (if the [DS\\_SETFONT](#) style was specified) and a [WM\\_INITDIALOG](#) message, and then the dialog box is displayed.

The [CreateDialogIndirect](#) function returns immediately after creating the dialog box.

To make the dialog box appear in the owner window upon being created, use the [WS\\_VISIBLE](#) style in the dialog box template.

Use the [DestroyWindow](#) function to destroy a dialog box created by the [CreateDialogIndirect](#) function.

A dialog box can contain up to 255 controls.

### Example

The following example uses the [CreateDialogIndirect](#) function to create a dialog box from a dialog box template in memory:

```
DLGPROC dlgprc = (DLGPROC) MakeProcInstance(DialogProc, hinst);
HWND hdlg;
BYTE FAR* lpbDlgTemp;

. /* Allocate global memory and build a dialog box template. */
.

hdlg = CreateDialogIndirect(hinst, lpbDlgTemp, hwndParent, dlgprc);
```

### See Also

[CreateDialog](#), [CreateDialogIndirectParam](#), [CreateDialogParam](#), [DestroyWindow](#), [MakeProcInstance](#), [WM\\_INITDIALOG](#), [WM\\_SETFONT](#)

## CreateDialogIndirectParam (3.0)

```
HWND CreateDialogIndirectParam(hinst, lpbDlgTmp, hwndOwner, dlgproc, lParamInit)
HINSTANCE hinst; /* handle of application instance */
const void FAR* lpbDlgTmp;
/* address of dialog box template
/
HWND hwndOwner;
/* handle of owner window
/
DLGPROC dlgproc;
/* instance address of dialog box procedure
/
LPARAM lParamInit;
/* initialization value
/
```

The CreateDialogIndirectParam function creates a modeless dialog box from a dialog box template in memory. Before displaying the dialog box, the function passes an application-defined value to the dialog box procedure as the lParam parameter of the WM\_INITDIALOG message. An application can use this value to initialize dialog box controls.

| Parameter  | Description                                                                                                                                                                                                                                                                                                 |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst      | Identifies the instance of the module that will create the dialog box.                                                                                                                                                                                                                                      |
| lpbDlgTmp  | Points to a global memory object that contains a dialog box template used to create the dialog box. This template is in the form of a DialogBoxHeader structure. For more information about this structure, see the <u>Dialog Box Resource</u> topic.                                                       |
| hwndOwner  | Identifies the window that owns the dialog box.                                                                                                                                                                                                                                                             |
| dlgproc    | Specifies the procedure-instance address of the dialog box procedure. The address must be created by using the <u>MakeProcInstance</u> function, except when the function and dialog box procedure are used in a DLL. For more information, see the description of the <u>DialogProc</u> callback function. |
| lParamInit | Specifies the value to pass to the dialog box when processing the <u>WM_INITDIALOG</u> message.                                                                                                                                                                                                             |

### Returns

The return value is the window handle of the dialog box if the function is successful. Otherwise, it is NULL.

### Comments

The CreateWindowEx function is called to create the dialog box. The dialog box procedure then receives a WM\_SETFONT message (if the DS\_SETFONT style was specified) and a WM\_INITDIALOG message, and then the dialog box is displayed.

The CreateDialogIndirectParam function returns immediately after creating the dialog box.

To make the dialog box appear in the owner window upon being created, use the WS\_VISIBLE style in the dialog box template.

Use the DestroyWindow function to destroy a dialog box created by the CreateDialogIndirectParam function.

A dialog box can contain up to 255 controls.

### Example

The following example calls the CreateDialogIndirectParam function to create a modeless dialog box from a dialog box template in memory. The example uses the lParamInit parameter to send two initialization parameters, wInitParm1 and wInitParm2, to the dialog box procedure when the WM\_INITDIALOG message is being processed.

```
#define MEM_LENGTH 100
HGLOBAL hglbDlgTemp;
BYTE FAR* lpbDlgTemp;
DLGPROC dlgproc = (DLGPROC) MakeProcInstance(DialogProc, hinst);
HWND hwndDlg;
```



```
/* Allocate a global memory object for the dialog box template. */
hglbDlgTemp = GlobalAlloc(GHND, MEM_LENGTH);
    . /* Build a DLGTEMPLATE structure in the memory object. */
    .
lpbDlgTemp = GlobalLock(hglbDlgTemp);
hwndDlg = CreateDialogIndirectParam(hinst, lpbDlgTemp,
    hwndParent, dlgprc, 0);
```

See Also

CreateDialog, CreateDialogIndirect, CreateDialogParam, DestroyWindow, MakeProcInstance,  
WM\_INITDIALOG, WM\_SETFONT

## CreateDialogParam (3.0)

```
HWND CreateDialogParam(hinst, lpszDlgTemp, hwndOwner, dlgproc, lParamInit)
HINSTANCE hinst;          /* handle of application instance */
LPCSTR lpszDlgTemp;
/* address of name of dialog box template
/
HWND hwndOwner;
/* handle of owner window
/
DLGPROC dlgproc;
/* instance address of dialog box procedure
/
LPARAM lParamInit;
/* initialization value
/
```

The CreateDialogParam function creates a modeless dialog box from a dialog box template resource. Before displaying the dialog box, the function passes an application-defined value to the dialog box procedure as the lParam parameter of the WM\_INITDIALOG message. An application can use this value to initialize dialog box controls.

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst       | Identifies an instance of the module whose executable file contains the dialog box template.                                                                                                                                                                                                                                               |
| lpszDlgTemp | Points to a null-terminated string that names the dialog box template.                                                                                                                                                                                                                                                                     |
| hwndOwner   | Identifies the window that owns the dialog box.                                                                                                                                                                                                                                                                                            |
| dlgproc     | Specifies the procedure-instance address of the dialog box procedure. The address must be created by using the <u>MakeProcInstance</u> function, except when the function and dialog box procedure are used in a DLL. For more information about the dialog box procedure, see the description of the <u>DialogProc</u> callback function. |
| lParamInit  | Specifies the value to pass to the dialog box when processing the <u>WM_INITDIALOG</u> message.                                                                                                                                                                                                                                            |

### Returns

The return value is the handle of the dialog box that was created, if the function is successful. Otherwise, it is NULL.

### Comments

The CreateWindowEx function is called to create the dialog box. The dialog box procedure then receives a WM\_SETFONT message (if the DS\_SETFONT style was specified) and a WM\_INITDIALOG message, and then the dialog box is displayed.

The CreateDialogParam function returns immediately after creating the dialog box.

To make the dialog box appear in the owner window upon being created, use the WS\_VISIBLE style in the dialog box template.

A dialog box can contain up to 255 controls.

### Example

The following example uses the CreateDialogParam function to create a modeless dialog box. The function passes the application-defined flags MIXEDCASE and WHOLEWORD, which will be received by the dialog box as the lParam parameter of the WM\_INITDIALOG message.

```
HWND hwndChangeBox;
DLGPROC dlgproc = (DLGPROC) MakeProcInstance(ChangeDlgProc, hinst);

hwndChangeBox = CreateDialogParam(hinst, "dlgFindBox",
    hwndParent, dlgproc, MIXEDCASE | WHOLEWORD);
```

### See Also

CreateDialog, CreateDialogIndirect, CreateDialogIndirectParam, DestroyWindow, WM\_INITDIALOG

## CreateIcon (3.0)

```
HICON CreateIcon(hinst, nWidth, nHeight, bPlanes, bBitsPixel, lpvANDbits, lpvXORbits)
HINSTANCE hinst; /* handle of application instance */
int nWidth; /* icon width */
int nHeight; /* icon height */
BYTE bPlanes; /* number of planes in XOR mask */
BYTE bBitsPixel; /* number of bits per pixel in XOR mask */
const void FAR* lpvANDbits; /* address of AND mask array */
const void FAR* lpvXORbits; /* address of XOR mask array */
```

The CreateIcon function creates an icon that has the specified width, height, colors, and bit patterns.

| Parameter  | Description                                                                                                                                                         |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst      | Identifies an instance of the module that will create the icon.                                                                                                     |
| nWidth     | Specifies the width, in pixels, of the icon.                                                                                                                        |
| nHeight    | Specifies the height, in pixels, of the icon.                                                                                                                       |
| bPlanes    | Specifies the number of planes in the XOR mask of the icon.                                                                                                         |
| bBitsPixel | Specifies the number of bits per pixel in the XOR mask of the icon.                                                                                                 |
| lpvANDbits | Points to an array of bytes that contains the bit values for the AND mask of the icon. This array must specify a monochrome mask.                                   |
| lpvXORbits | Points to an array of bytes that contains the bit values for the XOR mask of the icon. These bits can be the bits of a monochrome or device-dependent color bitmap. |

### Returns

The return value is the handle of the icon if the function is successful. Otherwise, it is NULL.

### Comments

The nWidth and nHeight parameters must specify a width and height supported by the current display driver, since the system cannot create icons of other sizes. An application can determine the width and height supported by the display driver by calling the [GetSystemMetrics](#) function, specifying the [SM\\_CXICON](#) or [SM\\_CYICON](#) constant.

Before terminating, an application must call the [DestroyIcon](#) function to free system resources associated with the icon.

### See Also

[DestroyIcon](#), [GetSystemMetrics](#)

## CreateMenu (2.x)

HMENU CreateMenu(void)

The CreateMenu function creates a menu. The menu is initially empty but can be filled with menu items by using the [AppendMenu](#) or [InsertMenu](#) function.

### Returns

The return value is the handle of the newly created menu if the function is successful. Otherwise, it is NULL.

### Comments

If the menu is not assigned to a window, an application must free system resources associated with the menu before exiting. An application frees menu resources by calling the [DestroyMenu](#) function. Windows automatically frees resources associated with a menu that is assigned to a window.

### Example

The following example creates a main menu and a pop-up menu and associates the pop-up menu with an item in the main menu:

```
HMENU hmenu;  
HMENU hmenuPopup;  
  
/* Create the main and pop-up menu handles. */  
  
hmenu = CreateMenu();  
hmenuPopup = CreatePopupMenu();  
  
/* Create the pop-up menu items. */  
  
AppendMenu(hmenuPopup, MF\_ENABLED | MF\_STRING, IDM_NEW,  
    "&New");  
AppendMenu(hmenuPopup, MF\_ENABLED | MF\_STRING, IDM_SAVE,  
    "&Save");  
AppendMenu(hmenuPopup, MF\_ENABLED | MF\_STRING, IDM_SAVE_AS,  
    "&Save As");  
  
/* Add the pop-up menu to the main menu. */  
  
AppendMenu(hmenu, MF\_ENABLED | MF\_POPUP, (UINT) hmenuPopup,  
    "&File");
```

### See Also

[AppendMenu](#), [DestroyMenu](#), [InsertMenu](#), [SetMenu](#)

## CreatePopupMenu (3.0)

HMENU CreatePopupMenu(void)

The CreatePopupMenu function creates an empty pop-up menu.

### Returns

The return value is the handle of the newly created menu if the function is successful. Otherwise, it is NULL.

### Comments

An application adds items to the pop-up menu by calling the [InsertMenu](#) and [AppendMenu](#) functions. The application can add the pop-up menu to an existing menu or pop-up menu, or it can display and track selections on the pop-up menu by calling the [TrackPopupMenu](#) function.

Before exiting, an application must free system resources associated with a pop-up menu if the menu is not assigned to a window. An application frees a menu by calling the [DestroyMenu](#) function.

### Example

The following example creates a main menu and a pop-up menu, and associates the pop-up menu with an item in the main menu:

```
HMENU hmenu;  
HMENU hmenuPopup;  
  
/* Create the main and pop-up menu handles. */  
  
hmenu = CreateMenu();  
hmenuPopup = CreatePopupMenu();  
  
/* Create the pop-up menu items. */  
  
AppendMenu(hmenuPopup, MF_ENABLED | MF_STRING, IDM_NEW,  
            "&New");  
AppendMenu(hmenuPopup, MF_ENABLED | MF_STRING, IDM_SAVE,  
            "&Save");  
AppendMenu(hmenuPopup, MF_ENABLED | MF_STRING, IDM_SAVE_AS,  
            "&Save As");  
  
/* Add the pop-up menu to the main menu. */  
  
AppendMenu(hmenu, MF_ENABLED | MF_POPUP, (UINT) hmenuPopup,  
            "&File");
```

### See Also

[AppendMenu](#), [CreateMenu](#), [InsertMenu](#), [SetMenu](#), [TrackPopupMenu](#)

## ■ CreateWindow (2.x)

```

HWND CreateWindow(lpszClassName, lpszWindowName, dwStyle, x, y, nWidth, nHeight,
                  hwndParent, hmenu, hinst, lpvParam)
LPCSTR lpszClassName; /* address of registered class name */
LPCSTR lpszWindowName;
/* address of window text
/
DWORD dwStyle;
/* window style
/
int x;
/* horizontal position of window
/
int y;
/* vertical position of window
/
int nWidth;
/* window width
/
int nHeight;
/* window height
/
HWND hwndParent;
/* handle of parent window
/
HMENU hmenu;
/* handle of menu or child-window identifier
/
HINSTANCE hinst;
/* handle of application instance
/
void FAR* lpvParam;
/* address of window-creation data
/

```

The CreateWindow function creates an overlapped, pop-up, or child window. The CreateWindow function specifies the window class, window title, window style, and (optionally) the initial position and size of the window. The CreateWindow function also specifies the window's parent (if any) and menu.

| Parameter      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszClassName  | Points to a null-terminated string specifying the window class. The class name can be any name registered with the RegisterClass function or any of the predefined control-class names. (See <a href="#">Control classes</a> ).                                                                                                                                                                                                                                                                                                                                                                                 |
| lpszWindowName | Points to a null-terminated string that represents the window name.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| dwStyle        | Specifies the style of window being created. This parameter can be a combination of the window styles and control styles given in the following Comments section.                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| x              | Specifies the initial x-position of the window. For an overlapped or pop-up window, the x parameter is the initial x-coordinate of the window's upper-left corner, in screen coordinates. For a child window, x is the x-coordinate of the upper-left corner of the window in the client area of its parent window.<br>If this value is CW_USEDEFAULT, Windows selects the default position for the window's upper-left corner and ignores the y parameter. CW_USEDEFAULT is valid only for overlapped windows. If CW_USEDEFAULT is specified for a non-overlapped window, the x and y parameters are set to 0. |
| y              | Specifies the initial y-position of the window. For an overlapped window, the y parameter is the initial y-coordinate of the window's upper-left corner. For a pop-up window, y is the y-coordinate, in screen coordinates, of the upper-left corner of the pop-up window. For list-box controls, y is the y-coordinate of the upper-left corner of the control's client area. For a child window, y is the y-coordinate of the upper-left corner of the child window. All of these coordinates are for the window, not the window's client area.                                                               |

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nWidth     | <p>If an overlapped window is created with the <u>WS_VISIBLE</u> style and the x parameter set to <u>CW_USEDEFAULT</u>, Windows ignores the y parameter.</p> <p>Specifies the width, in device units, of the window. For overlapped windows, the nWidth parameter is either the window's width (in screen coordinates) or <u>CW_USEDEFAULT</u>. If nWidth is <u>CW_USEDEFAULT</u>, Windows selects a default width and height for the window (the default width extends from the initial x-position to the right edge of the screen, and the default height extends from the initial y-position to the top of the icon area). <u>CW_USEDEFAULT</u> is valid only for overlapped windows. If <u>CW_USEDEFAULT</u> is specified in nWidth for a non-overlapped window, nWidth and nHeight are set to 0.</p> |
| nHeight    | <p>Specifies the height, in device units, of the window. For overlapped windows, the nHeight parameter is the window's height in screen coordinates. If the nWidth parameter is <u>CW_USEDEFAULT</u>, Windows ignores nHeight.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| hwndParent | <p>Identifies the parent or owner window of the window being created. A valid window handle must be supplied when creating a child window or an owned window. An owned window is an overlapped window that is destroyed when its owner window is destroyed, hidden when its owner is minimized, and that is always displayed on top of its owner window. For pop-up windows, a handle can be supplied but is not required. If the window does not have a parent window or is not owned by another window, the hwndParent parameter must be set to <u>HWND_DESKTOP</u>.</p>                                                                                                                                                                                                                                |
| hmenu      | <p>Identifies a menu or a child window. This parameter's meaning depends on the window style. For overlapped or pop-up windows, the hmenu parameter identifies the menu to be used with the window. It can be <u>NULL</u>, if the class menu is to be used. For child windows, hmenu identifies the child window and is an integer value that is used by a dialog box control to notify its parent of events (such as the <u>EN_HSCROLL</u> message). The child window identifier is determined by the <u>application</u> and should be unique for all child windows with the same parent window.</p>                                                                                                                                                                                                     |
| hinst      | <p>Identifies the instance of the module to be associated with the window.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| lpvParam   | <p>Points to a value that is passed to the window through the <u>CREATESTRUCT</u> structure referenced by the <u>lParam</u> parameter of the <u>WM_CREATE</u> message. If an application is calling <u>CreateWindow</u> to create a <u>multiple document interface (MDI)</u> client window, lpvParam must point to a <u>CLIENTCREATESTRUCT</u> structure.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                             |

#### Returns

The return value is the handle of the new window if the function is successful. Otherwise, it is NULL.

#### Comments

For overlapped, pop-up, and child windows, the CreateWindow function sends WM\_CREATE, WM\_GETMINMAXINFO, and WM\_NCCREATE messages to the window. If the WS\_VISIBLE style is specified, CreateWindow sends the window all the messages required to activate and show the window.

If the window style specifies a title bar, the window title pointed to by the lpszWindowName parameter is displayed in the title bar. When using CreateWindow to create controls such as buttons, check boxes, and edit controls, use the lpszWindowName parameter to specify the text of the control.

Before returning, the CreateWindow function sends a WM\_CREATE message to the window procedure.

Following are the predefined control classes an application can specify in the lpszClassName parameter:

| Class    | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BUTTON   | Designates a small rectangular child window that represents a button the user can turn on or off by clicking. Button controls can be used alone or in groups, and can either be labeled or appear without text. Button controls typically change appearance when the user clicks them.                                                                                                                                                                                                                                                                                                                                        |
| COMBOBOX | <p>Designates a control consisting of a list box and a selection field similar to an edit control. The list box may be displayed at all times or may be dropped down when the user selects a pop-up list box next to the selection field.</p> <p>Depending on the style of the combo box, the user can or cannot edit the contents of the selection field. If the list box is visible, typing characters into the selection box will cause the first list box entry that matches the characters typed to be highlighted. Conversely, selecting an item in the list box displays the selected text in the selection field.</p> |

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EDIT      | <p>Designates a rectangular child window in which the user can type text from the keyboard. The user selects the control, and gives it the input focus by clicking it or moving to it by pressing the TAB key. The user can type text when the control displays a flashing caret. The mouse can be used to move the cursor and select characters to be replaced, or to position the cursor for inserting characters. The BACKSPACE key can be used to delete characters.</p> <p>Edit controls use the variable-pitch System font and display characters from the Windows character set. Applications compiled to run with earlier versions of Windows display text with a fixed-pitch System font unless they have been marked by the Windows 3.0 MARK utility (with the MEMORY FONT option specified). An application can also send the <u>WM_SETFONT</u> message to the edit control to change the default font.</p> <p>Edit controls expand tab characters into as many space characters as are required to move the cursor to the next tab stop. Tab stops are assumed to be at every eighth character position.</p> |
| LISTBOX   | <p>Designates a list of character strings. This control is used whenever an application must present a list of names, such as filenames, from which the user can choose. The user can select a string by pointing to it and clicking. When a string is selected, it is highlighted and a notification message is passed to the parent window. A vertical or horizontal scroll bar can be used with a list box control to scroll lists that are too long for the control window. The list box automatically hides or shows the scroll bar as needed.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| MDICLIENT | <p>Designates an MDI client window. The MDI client window receives messages that control the MDI application's child windows. The recommended style bits are <u>WS_CLIPCHILDREN</u> and <u>WS_CHILD</u>. To create a scrollable MDI client window that allows the user to scroll MDI child windows into view, an application can also use the <u>WS_HSCROLL</u> and <u>WS_VSCROLL</u> styles.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| SCROLLBAR | <p>Designates a rectangle that contains a scroll box (also called a "thumb") and has direction arrows at both ends. The scroll bar sends a notification message to its parent window whenever the user clicks the control. The parent window is responsible for updating the position, if necessary. Scroll bar controls have the same appearance and function as scroll bars used in ordinary windows. Unlike scroll bars, however, scroll bar controls can be positioned anywhere in a window and used whenever needed to provide scrolling input for a window.</p> <p>The scroll bar class also includes size box controls (Maximize and Minimize buttons). These controls are small rectangles that the user can click to change the size of the window.</p>                                                                                                                                                                                                                                                                                                                                                         |
| STATIC    | <p>Designates a simple text field, box, or rectangle that can be used to label, box, or separate other controls. Static controls take no input and provide no output.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

Following are the window styles an application can specify in the dwStyle parameter.

| Style               | Meaning                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MDIS_ALLCHILDSTYLES | Creates an MDI client window that can have any combination of window styles. When this style is not specified, an MDI child window has the <u>WS_MINIMIZE</u> , <u>WS_MAXIMIZE</u> , <u>WS_HSCROLL</u> , and <u>WS_VSCROLL</u> styles as default settings.                                                                                                                                 |
| WS_BORDER           | Creates a window that has a border.                                                                                                                                                                                                                                                                                                                                                        |
| WS_CAPTION          | Creates a window that has a title bar (implies the <u>WS_BORDER</u> style). This style cannot be used with the <u>WS_DLGFRAAME</u> style.                                                                                                                                                                                                                                                  |
| WS_CHILD            | Creates a child window. Cannot be used with the <u>WS_POPUP</u> style.                                                                                                                                                                                                                                                                                                                     |
| WS_CHILDWINDOW      | Same as the <u>WS_CHILD</u> style.                                                                                                                                                                                                                                                                                                                                                         |
| WS_CLIPCHILDREN     | Excludes the area occupied by child windows when drawing within the parent window. Used when creating the parent window.                                                                                                                                                                                                                                                                   |
| WS_CLIPSIBLINGS     | Clips child windows relative to each other; that is, when a particular child window receives a paint message, the <u>WS_CLIPSIBLINGS</u> style clips all other overlapped child windows out of the region of the child window to be updated. (If <u>WS_CLIPSIBLINGS</u> is not specified and child windows overlap, it is possible, when drawing within the client area of a child window, |



|                            |                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                            | to draw within the client area of a neighboring child window.) For use with the <u>WS_CHILD</u> style only.                                                                                                                                                                                                                                                                                                                 |
| <u>WS_DISABLED</u>         | Creates a window that is initially disabled.                                                                                                                                                                                                                                                                                                                                                                                |
| <u>WS_DLDFRAME</u>         | Creates a window with a double border but no title.                                                                                                                                                                                                                                                                                                                                                                         |
| <u>WS_GROUP</u>            | Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls defined with the <u>WS_GROUP</u> style after the first control belong to the same group. The next control with the <u>WS_GROUP</u> style ends the style group and starts the next group (that is, one group ends where the next begins). Only dialog boxes use this style. |
| <u>WS_HSCROLL</u>          | Creates a window that has a horizontal scroll bar.                                                                                                                                                                                                                                                                                                                                                                          |
| <u>WS_MAXIMIZE</u>         | Creates a window of maximum size.                                                                                                                                                                                                                                                                                                                                                                                           |
| <u>WS_MAXIMIZEBOX</u>      | Creates a window that has a Maximize button.                                                                                                                                                                                                                                                                                                                                                                                |
| <u>WS_MINIMIZE</u>         | Creates a window that is initially minimized. For use with the <u>WS_OVERLAPPED</u> style only.                                                                                                                                                                                                                                                                                                                             |
| <u>WS_MINIMIZEBOX</u>      | Creates a window that has a Minimize button.                                                                                                                                                                                                                                                                                                                                                                                |
| <u>WS_OVERLAPPED</u>       | Creates an overlapped window. An overlapped window has a title and a border.                                                                                                                                                                                                                                                                                                                                                |
| <u>WS_OVERLAPPEDWINDOW</u> | Creates an overlapped window having the <u>WS_OVERLAPPED</u> , <u>WS_CAPTION</u> , <u>WS_SYSMENU</u> , <u>WS_THICKFRAME</u> , <u>WS_MINIMIZEBOX</u> , and <u>WS_MAXIMIZEBOX</u> styles.                                                                                                                                                                                                                                     |
| <u>WS_POPUP</u>            | Creates a pop-up window. Cannot be used with the <u>WS_CHILD</u> style.                                                                                                                                                                                                                                                                                                                                                     |
| <u>WS_POPUPWINDOW</u>      | Creates a pop-up window that has the <u>WS_BORDER</u> , <u>WS_POPUP</u> , and <u>WS_SYSMENU</u> styles. The <u>WS_CAPTION</u> style must be combined with the <u>WS_POPUPWINDOW</u> style to make the System menu visible.                                                                                                                                                                                                  |
| <u>WS_SYSMENU</u>          | Creates a window that has a System-menu box in its title bar. Used only for windows with title bars.                                                                                                                                                                                                                                                                                                                        |
| <u>WS_TABSTOP</u>          | Specifies one of any number of controls through which the user can move by using the TAB key. The TAB key moves the user to the next control specified by the <u>WS_TABSTOP</u> style. Only dialog boxes use this style.                                                                                                                                                                                                    |
| <u>WS_THICKFRAME</u>       | Creates a window with a thick frame that can be used to size the window.                                                                                                                                                                                                                                                                                                                                                    |
| <u>WS_VISIBLE</u>          | Creates a window that is initially visible. This applies to overlapped, child, and pop-up windows. For overlapped windows, the y parameter is used as a <u>ShowWindow</u> function parameter.                                                                                                                                                                                                                               |
| <u>WS_VSCROLL</u>          | Creates a window that has a vertical scroll bar.                                                                                                                                                                                                                                                                                                                                                                            |

The following styles may also be specified in the dwStyle parameter when a predefined control is being created:

- Button styles
- Combination box styles
- Edit control styles
- List box styles
- Scroll bar styles
- Static control styles

Following are the dialog box styles an application can specify in the dwStyle parameter:

| Style                | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>DS_LOCALEEDIT</u> | Specifies that edit controls in the dialog box will use memory in the application's data segment. By default, all edit controls in dialog boxes use memory outside the application's data segment. This feature may be suppressed by adding the <u>DS_LOCALEEDIT</u> flag to the Style command for the dialog box. If this flag is not used, <u>EM_GETHANDLE</u> and <u>EM_SETHANDLE</u> messages must not be used, because the storage for the control is not in the application's data segment. This feature does not affect edit controls created outside of dialog boxes. |

DS\_MODALFRAME      Creates a dialog box with a modal dialog box frame that can be combined with a title bar and System menu by specifying the WS\_CAPTION and WS\_SYSMENU styles.

DS\_NOIDLEMSG      Suppresses WM\_ENTERIDLE messages that Windows would otherwise send to the owner of the dialog box while the dialog box is displayed.

DS\_SYSMODAL      Creates a system-modal dialog box.

See Also

AnsiToOem, GetDialogBaseUnits, ShowWindow, CREATESTRUCT, CLIENTCREATESTRUCT

## Windows 3.1 changes

The following control styles have been added:

| Value               | Meaning                                                                                                                                                                                                                                                                                                                       |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ES_READONLY         | Prevents the user from entering or editing text in the edit control.                                                                                                                                                                                                                                                          |
| ES_WANTRETURN       | Specifies that a carriage return be inserted when the user presses the ENTER key while entering text into multiple-line edit control in a dialog box. Without this style, pressing the ENTER key has the same effect as pressing the dialog box's default pushbutton. This style has no effect on a single-line edit control. |
| CBS_DISABLENOSCROLL | The list box shows a disabled vertical scroll bar when the list box does not contain enough items to scroll. Without this style, the scroll bar is hidden when the list box does not contain enough items.                                                                                                                    |
| LBS_DISABLENOSCROLL | The list box shows a disabled vertical scroll bar when the list box does not contain enough items to scroll. Without this style, the scroll bar is hidden when the list box does not contain enough items.                                                                                                                    |

The SS\_USERITEM style has been removed.

### ■ CreateWindowEx (3.0)

```

HWND CreateWindowEx(dwExStyle, lpzClassName, lpzWindowName, dwStyle, x, y, nWidth,
                   nHeight, hwndParent, hmenu, hinst, lpvCreateParams)
DWORD dwExStyle;           /* extended window style */
LPCSTR lpzClassName;
/* address of registered class name
/
LPCSTR lpzWindowName;
/* address of window text
/
DWORD dwStyle;
/* window style
/
int x;
/* horizontal position of the window
/
int y;
/* vertical position of the window
/
int nWidth;
/* window width
/
int nHeight;
/* window height
/
HWND hwndParent;
/* handle of parent window
/
HMENU hmenu;
/* handle of menu or child-window identifier
/
HINSTANCE hinst;
/* handle of application instance
/
void FAR* lpvCreateParams;
/* address of window-creation data
/

```

The CreateWindowEx function creates an overlapped, pop-up, or child window with an extended style; otherwise, this function is identical to the CreateWindow function.

| Parameter | Description                                                                                    | Style                       | Meaning                                                                                                                                                                        |
|-----------|------------------------------------------------------------------------------------------------|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dwExStyle | Specifies the extended style of the window. This parameter can be one of the following values: | <u>WS_EX_ACCEPTFILES</u>    | Specifies that a window created with this style accepts drag-drop files.                                                                                                       |
|           |                                                                                                | <u>WS_EX_DLGMODALFRAME</u>  | Designates a window with a double border that may (optionally) be created with a title bar by specifying the <u>WS_CAPTION</u> style flag in the dwStyle parameter.            |
|           |                                                                                                | <u>WS_EX_NOPARENTNOTIFY</u> | Specifies that a child window created by using this style will not send the <u>WM_PARENTNOTIFY</u> message to its parent window when the child window is created or destroyed. |
|           |                                                                                                | <u>WS_EX_TOPMOST</u>        | Specifies that a window created with this style should be placed above all non-topmost windows and stay above them even when the window is                                     |

deactivated. An application can use the SetWindowPos function to add or remove this attribute.

#### WS\_EX\_TRANSPARENT

Specifies that a window created with this style is to be transparent. That is, any windows that are beneath the window are not obscured by the window. A window created with this style receives WM\_PAINT messages only after all sibling windows beneath it have been updated.

|                 |                                                                                                                                                                                                                              |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpzClassName    | Points to a null-terminated string containing the name of the window class.                                                                                                                                                  |
| lpzWindowName   | Points to a null-terminated string containing the name of the window.                                                                                                                                                        |
| dwStyle         | Specifies the style of the window. For a list of the window styles that can be specified in this parameter, see the preceding description of the <u>CreateWindow</u> function.                                               |
| x               | Specifies the initial left-side position of the window.                                                                                                                                                                      |
| y               | Specifies the initial top position of the window.                                                                                                                                                                            |
| nWidth          | Specifies the width, in device units, of the window.                                                                                                                                                                         |
| nHeight         | Specifies the height, in device units, of the window.                                                                                                                                                                        |
| hwndParent      | Identifies the parent or owner window of the window to be created.                                                                                                                                                           |
| hmenu           | Identifies a menu or a child window. The meaning depends on the window style.                                                                                                                                                |
| hinst           | Identifies the instance of the module to be associated with the window.                                                                                                                                                      |
| lpvCreateParams | Contains any application-specific creation parameters. The window being created may access this data when the <u>CREATESTRUCT</u> structure is passed to the window by the <u>WM_NCCREATE</u> and <u>WM_CREATE</u> messages. |

#### Returns

The return value identifies the new window if the function is successful. Otherwise, it is NULL.

#### Comments

The CreateWindowEx function sends the following messages to the window being created:

WM\_NCCREATE  
WM\_NCCALCSIZE  
WM\_CREATE

#### Example

The following example creates a main window that has the WS\_EX\_TOPMOST extended style, makes the window visible, and updates the window's client area:

```
char szClassName[] = "MyClass";

/* Create the main window. */

hwnd = CreateWindowEx(WS_EX_TOPMOST, szClassName, "Grouper",
    WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT,
    CW_USEDEFAULT, CW_USEDEFAULT, NULL, NULL,
    hinst, NULL);

/* Make the window visible and update its client area. */

ShowWindow(hwnd, SW_SHOW); /* always show the window */
UpdateWindow(hwnd);
```

#### See Also

CreateWindow, SetWindowPos, CREATESTRUCT

## Windows 3.1 changes

The following styles may be used for the dwExStyle parameter:

| Style             | Meaning                                                                                                                                                                                                                                                                                |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WS_EX_ACCEPTFILES | Specifies that a window created with this style accepts drag-drop files.                                                                                                                                                                                                               |
| WS_EX_TOPMOST     | Specifies that a window created with this style should be placed above all non-topmost windows and stay above them even when the window is deactivated. An application can use the <u>SetWindowPos</u> function to add or remove this attribute.                                       |
| WS_EX_TRANSPARENT | Specifies that a window created with this style is to be transparent. That is, any windows that are beneath the window are not obscured by the window. A window created with this style receives <u>WM_PAINT</u> messages only after all sibling windows beneath it have been updated. |

WS\_EX\_ACCEPTFILES 0x0000010L

Specifies that a window created with this style accepts drag-drop files.

WS\_EX\_ACCEPTFILES 0x0000010L



WS\_EX\_DLGMODALFRAME 0x0000001L

Designates a window with a double border that may (optionally) be created with a title bar by specifying the WS\_CAPTION style flag in the dwStyle parameter.

WS\_EX\_DLGMODALFRAME 0x0000001L

WS\_EX\_NOPARENTNOTIFY 0x00000004L

Specifies that a child window created by using this style will not send the WM\_PARENTNOTIFY message to its parent window when the child window is created or destroyed.

WS\_EX\_NOPARENTNOTIFY 0x00000004L

WS\_EX\_TOPMOST 0x00000008L

Specifies that a window created with this style should be placed above all non-topmost windows and stay above them even when the window is deactivated. An application can use the SetWindowPos function to add or remove this attribute.

WS\_EX\_TOPMOST 0x00000008L

WS\_EX\_TRANSPARENT 0x00000020L

Specifies that a window created with this style is to be transparent. That is, any windows that are beneath the window are not obscured by the window. A window created with this style receives WM\_PAINT messages only after all sibling windows beneath it have been updated.

WS\_EX\_TRANSPARENT 0x0000020L



## DefDlgProc (3.0)

```
LRESULT DefDlgProc(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)
/* handle of dialog box */
/* message */
/* first message parameter */
/* second message parameter */
```

The DefDlgProc function provides default processing for any Windows messages that a dialog box with a private window class does not process.

| Parameter | Description                                                    |
|-----------|----------------------------------------------------------------|
| hWndDlg   | Identifies the dialog box.                                     |
| uMsg      | Specifies the message to be processed.                         |
| wParam    | Specifies 16 bits of additional message-dependent information. |
| lParam    | Specifies 32 bits of additional message-dependent information. |

### Returns

The return value specifies the result of the message processing and depends on the message sent.

### Comments

The DefDlgProc function is the window procedure for the DIALOG window class. An application that creates new window classes that inherit dialog box functionality should use this function. DefDlgProc is not intended to be called as the default handler for messages within a dialog box procedure, since doing so will result in recursive execution.

An application creates a dialog box by calling one of the following functions:

| Function                         | Description                                                             |
|----------------------------------|-------------------------------------------------------------------------|
| <u>CreateDialog</u>              | Creates a modeless dialog box.                                          |
| <u>CreateDialogIndirect</u>      | Creates a modeless dialog box.                                          |
| <u>CreateDialogIndirectParam</u> | Creates a modeless dialog box and passes data to it when it is created. |
| <u>CreateDialogParam</u>         | Creates a modeless dialog box and passes data to it when it is created. |
| <u>DialogBox</u>                 | Creates a modal dialog box.                                             |
| <u>DialogBoxIndirect</u>         | Creates a modal dialog box.                                             |
| <u>DialogBoxIndirectParam</u>    | Creates a modal dialog box and passes data to it when it is created.    |
| <u>DialogBoxParam</u>            | Creates a modal dialog box and passes data to it when it is created.    |

### See Also

DefWindowProc

### DefDriverProc (3.1)

```
LRESULT DefDriverProc(dwDriverIdentifier, hdrv, uMsg, IParam1, IParam2)
DWORD dwDriverIdentifier;    /* installable-driver identifier */
HDRV hdrv;                   /* handle of
installable driver
/
UINT uMsg;
* message number
/
LPARAM IParam1;
* first message parameter
/
LPARAM IParam2;
* second message parameter
/
```

The DefDriverProc function provides default processing for any messages not processed by an installable driver.

| Parameter          | Description                                                                                                                    |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------|
| dwDriverIdentifier | Identifies an installable driver. This parameter must have been obtained by a previous call to the <u>OpenDriver</u> function. |
| hdrv               | Identifies the installable driver.                                                                                             |
| uMsg               | Specifies the message to be processed.                                                                                         |
| IParam1            | Specifies 32 bits of additional message-dependent information.                                                                 |
| IParam2            | Specifies 32 bits of additional message-dependent information.                                                                 |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The DefDriverProc function processes messages that are not handled by the DriverProc function.

#### See Also

OpenDriver, SendDriverMessage

### DeferWindowPos (3.0)

```

HDWP DeferWindowPos(hdwp, hwnd, hwndInsertAfter, x, y, cx, cy, flags)
HDWP hdwp; /* handle of internal structure */
HWND hwnd; /* handle of
window to position
/
HWND hwndInsertAfter;
* placement-order handle
/
int x;
* horizontal position
/
int y;
* vertical position
/
int cx;
* width
/
int cy;
* height
/
UINT flags;
* window-positioning flags
/

```

The DeferWindowPos function updates the given internal structure for the given window. The function then returns the handle of the updated structure. The EndDeferWindowPos function uses the information in this structure to change the position and size of a number of windows simultaneously.

| Parameter       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|----------------------------------------------|--------------|-------------------------------------------------------------------------------------------------------------------------|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hdwp            | Identifies an internal structure that contains size and position information for one or more windows. This structure is returned by the <u>BeginDeferWindowPos</u> function or by the most recent call to the DeferWindowPos function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |
| hwnd            | Identifies the window for which update information is to be stored in the structure.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |
| hwndInsertAfter | Identifies a window that will precede the positioned window in the Z-order. This parameter must be a window handle, or one of the following values: <table border="1" data-bbox="487 1218 1380 1596"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>HWND_BOTTOM</td> <td>Places the window at the bottom of the Z-order. If hwnd identifies a topmost window, the window loses its topmost status; the system places the window at the bottom of all other windows.</td> </tr> <tr> <td>HWND_TOP</td> <td>Places the window at the top of the Z-order.</td> </tr> <tr> <td>HWND_TOPMOST</td> <td>Places the window above all non-topmost windows. The window maintains its topmost position even when it is deactivated.</td> </tr> <tr> <td>HWND_NOTOPMOST</td> <td>Repositions the window to the top of all non-topmost windows (that is, behind all topmost windows). This flag has no effect if the window is already a non-topmost window.</td> </tr> </tbody> </table> <p>This parameter is ignored if the <u>SWP_NOZORDER</u> flag is set in the flags parameter.</p> | Value | Meaning | HWND_BOTTOM | Places the window at the bottom of the Z-order. If hwnd identifies a topmost window, the window loses its topmost status; the system places the window at the bottom of all other windows. | HWND_TOP | Places the window at the top of the Z-order. | HWND_TOPMOST | Places the window above all non-topmost windows. The window maintains its topmost position even when it is deactivated. | HWND_NOTOPMOST | Repositions the window to the top of all non-topmost windows (that is, behind all topmost windows). This flag has no effect if the window is already a non-topmost window. |
| Value           | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |
| HWND_BOTTOM     | Places the window at the bottom of the Z-order. If hwnd identifies a topmost window, the window loses its topmost status; the system places the window at the bottom of all other windows.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |
| HWND_TOP        | Places the window at the top of the Z-order.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |
| HWND_TOPMOST    | Places the window above all non-topmost windows. The window maintains its topmost position even when it is deactivated.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |
| HWND_NOTOPMOST  | Repositions the window to the top of all non-topmost windows (that is, behind all topmost windows). This flag has no effect if the window is already a non-topmost window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |
| x               | Specifies the x-coordinate of the window's upper-left corner.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |
| y               | Specifies the y-coordinate of the window's upper-left corner.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |
| cx              | Specifies the window's new width.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |
| cy              | Specifies the window's new height.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |
| flags           | Specifies one of eight possible 16-bit values that affect the size and position of the window. This parameter can be a combination of the following values: <table border="1" data-bbox="487 1848 1380 1890"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Value | Meaning |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |
| Value           | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |             |                                                                                                                                                                                            |          |                                              |              |                                                                                                                         |                |                                                                                                                                                                            |

|                        |                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>SWP_DRAWERFRAME</u> | Draws a frame (defined in the window's class description) around the window.                                                                                                                                                                                                                                                                                                                         |
| <u>SWP_HIDEWINDOW</u>  | Hides the window.                                                                                                                                                                                                                                                                                                                                                                                    |
| <u>SWP_NOACTIVATE</u>  | Does not activate the window.                                                                                                                                                                                                                                                                                                                                                                        |
| <u>SWP_NOMOVE</u>      | Retains current position (ignores the x and y parameters).                                                                                                                                                                                                                                                                                                                                           |
| <u>SWP_NOREDRAW</u>    | Does not redraw changes. If this flag is set, no repainting of any kind occurs. This applies to the client area, the non-client area (including the title and scroll bars), and any part of the parent window uncovered as a result of the moved window. When this flag is set, the application must explicitly invalidate or redraw any parts of the window and parent window that must be redrawn. |
| <u>SWP_NOSIZE</u>      | Retains current size (ignores the cx and cy parameters).                                                                                                                                                                                                                                                                                                                                             |
| <u>SWP_NOZORDER</u>    | Retains current ordering (ignores the hwndInsertAfter parameter).                                                                                                                                                                                                                                                                                                                                    |
| <u>SWP_SHOWWINDOW</u>  | Displays the window.                                                                                                                                                                                                                                                                                                                                                                                 |

#### Returns

The return value is a handle of the updated structure if the function is successful. This handle may differ from the one passed to the function as the `hwndp` parameter and should be passed to the next call to `DeferWindowPos` or to the `EndDeferWindowPos` function.

The return value is `NULL` if insufficient system resources are available for the function to complete successfully and the repositioning process is terminated.

#### Comments

If a call to `DeferWindowPos` fails, the application should abandon the window-positioning operation without calling the `EndDeferWindowPos` function.

If the `SWP_NOZORDER` flag is not specified, Windows places the window identified by the `hwnd` parameter in the position following the window identified by the `hwndInsertAfter` parameter. If `hwndInsertAfter` is `NULL`, Windows places the window identified by `hwnd` at the top of the list. If `hwndInsertAfter` is `HWND_BOTTOM`, Windows places the window identified by `hwnd` at the bottom of the list.

All coordinates for child windows are relative to the upper-left corner of the parent window's client area.

A window can be made a topmost window either by setting the `hwndInsertAfter` parameter to `HWND_TOPMOST` and ensuring that the `SWP_NOZORDER` flag is not set, or by setting a window's Z-order so that it is above any existing topmost windows. When a non-topmost window is made topmost, its owned windows are also made topmost. Its owners are not changed.

If neither `SWP_NOACTIVATE` nor `SWP_NOZORDER` is specified (that is, when the application requests that a window be simultaneously activated and placed in the specified Z-order), the value specified in `hwndInsertAfter` is used only in the following circumstances:

- Neither `HWND_TOPMOST` or `HWND_NOTOPMOST` is specified in the `hwndInsertAfter` parameter.
- The window specified in the `hwnd` parameter is not the active window.

An application cannot activate an inactive window without also bringing it to the top of the Z-order. Applications can change the Z-order of an activated window without restrictions or activate a window and then move it to the top of the topmost or non-topmost windows.

A topmost window is no longer topmost if it is repositioned to the bottom (`HWND_BOTTOM`) of the Z-order or after any non-topmost window. When a topmost window is made non-topmost, the window and all of its owners, and its owned windows, are also made non-topmost.

A non-topmost window may own a topmost window, but not vice versa. Any window (for example, a dialog box) owned by a topmost window is itself made topmost to ensure that all owned windows stay above their owner.

#### See Also

[BeginDeferWindowPos](#), [EndDeferWindowPos](#)

### Windows 3.1 changes

If the `hwndInsertAfter` parameter is `HWND_TOPMOST`, the system places the window identified by the `hwnd` parameter above all non-topmost windows. The window maintains its topmost position even when the window is deactivated. If the `hwndInsertAfter` parameter is `HWND_BOTTOM` and `hwnd` identifies a topmost window, the window loses its topmost status--the system places the window at the bottom of all other windows.

The following window-positioning flags are new for Windows version 3.1:

| Value                       | Meaning                                                                                                                                                                                                 |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>HWND_BOTTOM</code>    | Places the window at the bottom of the Z order. If <code>hwnd</code> identifies a topmost window, the window loses its topmost status--the system places the window at the bottom of all other windows. |
| <code>HWND_TOP</code>       | Places the window at the top of the Z order.                                                                                                                                                            |
| <code>HWND_TOPMOST</code>   | Places the window above all non-topmost windows. The window maintains its topmost position even when the window is deactivated.                                                                         |
| <code>HWND_NOTOPMOST</code> | Repositions the window to the top of all non-topmost windows (that is, behind all topmost window).                                                                                                      |

SWP\_DRAWFRAME SWP\_FRAMECHANGED

Draws a frame (defined in the window's class description) around the window.

SWP\_DRAWFRAME SWP\_FRAMECHANGED

SWP\_HIDEWINDOW 0x0080

Hides the window.



SWP\_HIDEWINDOW 0x0080

SWP\_NOACTIVATE 0x0010

Does not activate the window.

SWP\_NOACTIVATE 0x0010

SWP\_NOMOVE 0x0002

Retains current position (ignores the x and y parameters).

SWP\_NOMOVE 0x0002

SWP\_NOREDRAW 0x0008

Does not redraw changes. If this flag is set, no repainting of any kind occurs. This applies to the client area, the non-client area (including the title and scroll bars), and any part of the parent window uncovered as a result of the moved window. When this flag is set, the application must explicitly invalidate or redraw any parts of the window and parent window that must be redrawn.

SWP\_NOEDRAW 0x0008

SWP\_NOSIZE 0x0001

Retains current size (ignores the cx and cy parameters).



SWP\_NOSIZE 0x0001

SWP\_NOZORDER 0x0004

Retains current ordering (ignores the hWndInsertAfter parameter).

SWP\_NOZORDER 0x0004

SWP\_SHOWWINDOW 0x0040

Displays the window.

SWP\_SHOWWINDOW 0x0040

### DefFrameProc (3.0)

```
LRESULT DefFrameProc(hwnd, hwndMDIClient, uMsg, wParam, lParam)
HWND hwnd; /* handle of frame window */
HWND hwndMDIClient; /* handle of client window */
UINT uMsg; /* message */
WPARAM wParam; /* first message parameter */
LPARAM lParam; /* second message parameter */
```

The DefFrameProc function provides default processing for any Windows messages that the window procedure of a multiple document interface (MDI) frame window does not process. All window messages that are not explicitly processed by the window procedure must be passed to the DefFrameProc function, not the DefWindowProc function.

| Parameter     | Description                                                    |
|---------------|----------------------------------------------------------------|
| hwnd          | Identifies the MDI frame window.                               |
| hwndMDIClient | Identifies the MDI client window.                              |
| uMsg          | Specifies the message to be processed.                         |
| wParam        | Specifies 16 bits of additional message-dependent information. |
| lParam        | Specifies 32 bits of additional message-dependent information. |

#### Returns

The return value specifies the result of the message processing and depends on the message sent. If the hwndMDIClient parameter is NULL, the return value is the same as for the DefWindowProc function.

#### Comments

Typically, when an application's window procedure does not handle a message, it passes the message to the DefWindowProc function, which processes the message. MDI applications use the DefFrameProc and DefMDIChildProc functions instead of DefWindowProc to provide default message processing. All messages that an application would usually pass to DefWindowProc (such as nonclient messages and WM\_SETTEXT) should be passed to DefFrameProc instead. In addition to handling these messages, DefFrameProc also handles the following messages:

| Message     | Response                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WM_COMMAND  | The frame window of an MDI application receives the WM_COMMAND message to activate a particular MDI child window. The window identifier accompanying this message will identify the MDI child window assigned by Windows, starting with the first identifier specified by the application when it created the MDI client window. This value of the first identifier must not conflict with menu-item identifiers. |
| WM_MENUCHAR | When the user presses the ALT+ key combination, the System menu (often called Control menu) of the active MDI child window will be selected.                                                                                                                                                                                                                                                                      |
| WM_SETFOCUS | DefFrameProc passes focus on to the MDI client, which in turn passes the focus on to the active MDI child window.                                                                                                                                                                                                                                                                                                 |
| WM_SIZE     | If the frame window procedure passes this message to DefFrameProc, the MDI client window will be resized to fit in the new client area. If the frame window procedure sizes the MDI client to a different size, it should not pass the message to <u>DefWindowProc</u> .                                                                                                                                          |

#### See Also

DefMDIChildProc, DefWindowProc

## DefHookProc (2.x)

```
DWORD DefHookProc(nCode, uParam, dwParam, lphhook)
int nCode; /* process code */
UINT uParam; /* first
message parameter
/
DWORD dwParam;
* second message parameter
/
HHOOK FAR* lphhook;
* points to address of next hook function
/
```

This function is obsolete but has been retained for backward compatibility with Windows versions 3.0 and earlier. Applications written for Windows version 3.1 should use the [CallNextHookEx](#) function.

The DefHookProc function calls the next function in a chain of hook functions. A hook function is a function that processes events before they are sent to an application's message-processing loop in the WinMain function. When an application defines more than one hook function by using the [SetWindowsHook](#) function, Windows forms a linked list or hook chain. Windows places functions of the same type in a chain.

| Parameter | Description                                                                                                                                                            |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nCode     | Specifies a code used by the Windows hook function (also called the message-filter function) to determine how to process the message.                                  |
| uParam    | Specifies 16 bits of additional message-dependent information.                                                                                                         |
| dwParam   | Specifies 32 bits of additional message-dependent information.                                                                                                         |
| lphhook   | Points to the variable that contains the procedure-instance address of the previously installed hook function returned by the <a href="#">SetWindowsHook</a> function. |

### Returns

The return value specifies the result of the event processing and depends on the event.

### Comments

Windows changes the value at the location pointed to by the lphhook parameter after an application calls the [UnhookWindowsHook](#) function. For more information, see the description of the [UnhookWindowsHook](#) function.

### See Also

[CallNextHookEx](#), [SetWindowsHook](#), [UnhookWindowsHook](#)

### DefMDIChildProc (3.0)

```
LRESULT DefMDIChildProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
/* handle of child window */
/* message */
WPARAM wParam;
* first message parameter
/
LPARAM lParam;
* second message parameter
/
```

The DefMDIChildProc function provides default processing for any Windows messages that the window procedure of a multiple document interface (MDI) child window does not process. All window messages that are not explicitly processed by the window procedure must be passed to the DefMDIChildProc function, not the DefWindowProc function.

| Parameter | Description                                                    |
|-----------|----------------------------------------------------------------|
| hwnd      | Identifies the MDI child window.                               |
| uMsg      | Specifies the message to be processed.                         |
| wParam    | Specifies 16 bits of additional message-dependent information. |
| lParam    | Specifies 32 bits of additional message-dependent information. |

#### Returns

The return value specifies the result of the message processing and depends on the message sent.

#### Comments

This function assumes that the parent of the window identified by the hwnd parameter was created with the MDICLIENT class.

Typically, when an application's window procedure does not handle a message, it passes the message to the DefWindowProc function, which processes the message. MDI applications use the DefFrameProc and DefMDIChildProc functions instead of DefWindowProc to provide default message processing. All messages that an application would usually pass to DefWindowProc (such as nonclient messages and WM\_SETTEXT) should be passed to DefMDIChildProc instead. In addition to handling these messages, DefMDIChildProc also handles the following messages:

| Message          | Response                                                                                                                                                                                                                       |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WM_CHILDACTIVATE | Performs activation processing when child windows are sized, moved, or shown. This message must be passed.                                                                                                                     |
| WM_GETMINMAXINFO | Calculates the size of a maximized MDI child window based on the current size of the MDI client window.                                                                                                                        |
| WM_MENUCHAR      | Sends the keystrokes to the frame window.                                                                                                                                                                                      |
| WM_MOVE          | Recalculates MDI client scroll bars, if they are present.                                                                                                                                                                      |
| WM_SETFOCUS      | Activates the child window if it is not the active MDI child window.                                                                                                                                                           |
| WM_SIZE          | Performs necessary operations when changing the size of a window, especially when maximizing or restoring an MDI child window. Failing to pass this message to <u>DefMDIChildProc</u> will produce highly undesirable results. |
| WM_SYSCOMMAND    | Also handles the next window command.                                                                                                                                                                                          |

#### See Also

DefFrameProc, DefWindowProc



## DefWindowProc (2.x)

```
LRESULT DefWindowProc(hwnd, uMsg, wParam, lParam)
HWND hwnd;          /* handle of window */
UINT uMsg;          /* type of message */
/
WPARAM wParam;
* first message parameter
/
LPARAM lParam;
* second message parameter
/
```

The DefWindowProc function calls the default window procedure. The default window procedure provides default processing for any window messages that an application does not process. This function ensures that every message is processed. It should be called with the same parameters as those received by the window procedure.

| Parameter | Description                                                    |
|-----------|----------------------------------------------------------------|
| hwnd      | Identifies the window that received the message.               |
| uMsg      | Specifies the message.                                         |
| wParam    | Specifies 16 bits of additional message-dependent information. |
| lParam    | Specifies 32 bits of additional message-dependent information. |

### Returns

The return value is the result of the message processing and depends on the message sent.

### Example

The following example shows a typical window procedure. A switch statement is used to process individual messages. All messages not processed are passed on to the DefWindowProc function.

```
LRESULT CALLBACK MainWndProc(hwnd, msg, wParam, lParam)
HWND hwnd; /* handle of window */
UINT msg; /* type of message */
WPARAM wParam; /* additional information */
LPARAM lParam; /* additional information */
{
    switch (msg) {

        /*
         * Process whatever messages you want here and send the
         * rest to DefWindowProc.
         */

        default:
            return (DefWindowProc(hwnd, message, wParam, lParam));
    }
}
```

### See Also

[DefDlgProc](#)

## DeleteMenu (3.0)

```
BOOL DeleteMenu(hmenu, idItem, fuFlags)
HMENU hmenu;      /* handle of menu */
UINT idItem;
UINT fuFlags;
/* menu flags
/
```

```
/* menu-item identifier */
```

```
/
```

The DeleteMenu function deletes an item from a menu. If the menu item has an associated pop-up menu, DeleteMenu destroys the handle of the pop-up menu and frees the memory used by the pop-up menu.

| Parameter            | Description                                                                                                                                                                                                                                                                                                                |       |         |                     |                                                          |                      |                                                                                   |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---------------------|----------------------------------------------------------|----------------------|-----------------------------------------------------------------------------------|
| hmenu                | Identifies the menu to be deleted.                                                                                                                                                                                                                                                                                         |       |         |                     |                                                          |                      |                                                                                   |
| idItem               | Specifies the menu item to be deleted, as determined by the fuFlags parameter.                                                                                                                                                                                                                                             |       |         |                     |                                                          |                      |                                                                                   |
| fuFlags              | Specifies how the idItem parameter is interpreted. This parameter can be one of the following values:                                                                                                                                                                                                                      |       |         |                     |                                                          |                      |                                                                                   |
|                      | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>MF_BYCOMMAND</u></td><td>The idItem parameter specifies the menu-item identifier.</td></tr><tr><td><u>MF_BYPOSITION</u></td><td>The idItem parameter specifies the zero-based relative position of the menu item.</td></tr></tbody></table> | Value | Meaning | <u>MF_BYCOMMAND</u> | The idItem parameter specifies the menu-item identifier. | <u>MF_BYPOSITION</u> | The idItem parameter specifies the zero-based relative position of the menu item. |
| Value                | Meaning                                                                                                                                                                                                                                                                                                                    |       |         |                     |                                                          |                      |                                                                                   |
| <u>MF_BYCOMMAND</u>  | The idItem parameter specifies the menu-item identifier.                                                                                                                                                                                                                                                                   |       |         |                     |                                                          |                      |                                                                                   |
| <u>MF_BYPOSITION</u> | The idItem parameter specifies the zero-based relative position of the menu item.                                                                                                                                                                                                                                          |       |         |                     |                                                          |                      |                                                                                   |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

Whenever a menu changes (whether or not the menu is in a window that is displayed), the application should call the DrawMenuBar function.

### See Also

AppendMenu, CreateMenu, DrawMenuBar, InsertMenu, RemoveMenu

MF\_BYCOMMAND 0x0000

The `idItem` parameter specifies the menu-item identifier.

MF\_BYCOMMAND 0x0000

MF\_BYPOSITION 0x0400

The `idItem` parameter specifies the zero-based relative position of the menu item.

MF\_BYPOSITION 0x0400

## DestroyCaret (2.x)

void DestroyCaret(void)

The DestroyCaret function destroys the current caret shape, frees the caret from the window that currently owns it, and removes the caret from the screen if it is visible. The DestroyCaret function checks the ownership of the caret and destroys the caret only if a window in the current task owns it.

If the caret shape was previously a bitmap, DestroyCaret does not free the bitmap.

### Returns

This function does not return a value.

### Comments

The caret is a shared resource. If a window has created a caret shape, it should destroy that shape before it loses the input focus or becomes inactive.

### See Also

[CreateCaret](#), [HideCaret](#), [ShowCaret](#)

## DestroyCursor (3.0)

```
BOOL DestroyCursor(hcur)
HCURSOR hcur;      /* handle of cursor to destroy */
```

The DestroyCursor function destroys a cursor that was previously created by the CreateCursor or LoadCursor function and frees any memory that the cursor occupied.

| Parameter | Description                                                                   |
|-----------|-------------------------------------------------------------------------------|
| hcur      | Identifies the cursor to be destroyed. The cursor must not be in current use. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### See Also

CreateCursor, CreateIcon, DestroyIcon, LoadCursor



## DestroyIcon (3.0)

```
BOOL DestroyIcon(hicon)
HICON hicon;      /* handle of icon to destroy */
```

The DestroyIcon function destroys an icon that was created by the CreateIcon or LoadIcon function and frees any memory that the icon occupied.

| Parameter | Description                          |
|-----------|--------------------------------------|
| hicon     | Identifies the icon to be destroyed. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### See Also

CreateCursor, CreateIcon, DestroyCursor, LoadIcon

## DestroyMenu (2.x)

```
BOOL DestroyMenu(HMENU hmenu)
/* handle of menu to destroy */
```

The DestroyMenu function destroys a menu and frees any memory that the menu occupied.

| Parameter | Description                          |
|-----------|--------------------------------------|
| hmenu     | Identifies the menu to be destroyed. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### See Also

[CreateMenu](#)

## DestroyWindow (2.x)

```
BOOL DestroyWindow(hwnd)
HWND hwnd;    /* handle of window to destroy */
```

The DestroyWindow function destroys the specified window. The function sends appropriate messages to the window to deactivate it and remove the input focus. It also destroys the window's menu, flushes the application queue, destroys outstanding timers, removes clipboard ownership, and breaks the clipboard-viewer chain (if the window is at the top of the viewer chain). It sends WM\_DESTROY and WM\_NCDESTROY messages to the window.

If the given window is the parent of any windows, DestroyWindow automatically destroys these child windows when it destroys the parent window. The function destroys child windows first, and then the window itself.

The DestroyWindow function also destroys modeless dialog boxes created by the CreateDialog function.

| Parameter | Description                            |
|-----------|----------------------------------------|
| hwnd      | Identifies the window to be destroyed. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

Applications should always call the DestroyWindow function to destroy their top-level windows before terminating.

If the window being destroyed is a child window and does not have the WS\_NOPARENTNOTIFY style set, a WM\_PARENTNOTIFY message is sent to the parent.

### Example

The following example responds to the application-defined menu command IDM\_EXIT, and then calls DestroyWindow to destroy the window:

```
case IDM_EXIT:
    DestroyWindow(hwnd);
    return 0;
```

### See Also

CreateDialog, CreateWindow, CreateWindowEx, WM\_DESTROY, WM\_NCDESTROY, WM\_PARENTNOTIFY

## DialogBox (2.x)

```
int DialogBox(hinst, lpszDlgTemp, hwndOwner, dlgprc)
HINSTANCE hinst;          /* handle of application instance */
LPCSTR lpszDlgTemp;
/* address of dialog box template name
/
HWND hwndOwner;
/* handle of owner window
/
DLGPROC dlgprc;
/* instance address of dialog box procedure
/
```

The DialogBox function creates a modal dialog box from a dialog box template resource.

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst       | Identifies an instance of the module whose executable file contains the dialog box template.                                                                                                                                                                                                                                                                 |
| lpszDlgTemp | Points to a null-terminated string that names the dialog box template.                                                                                                                                                                                                                                                                                       |
| hwndOwner   | Identifies the window that owns the dialog box.                                                                                                                                                                                                                                                                                                              |
| dlgprc      | Specifies the procedure-instance address of the dialog box procedure. The address must be created by using the <a href="#">MakeProcInstance</a> function, except when the function and dialog box procedure are used in a DLL. For more information about the dialog box procedure, see the description of the <a href="#">DialogProc</a> callback function. |

### Returns

The return value specifies the value of the `nResult` parameter specified in the [EndDialog](#) function that is used to terminate the dialog box. The system processes values returned by the dialog box procedure and does not return them to the application. The return value is -1 if the function cannot create the dialog box.

### Comments

The [CreateWindowEx](#) function is called to create the dialog box. The dialog box procedure then receives a [WM\\_SETFONT](#) message (if `DS_SETFONT` style was specified) and a [WM\\_INITDIALOG](#) message, and then the dialog box is displayed.

The DialogBox function does not return control until the dialog box procedure terminates the modal dialog box by calling the [EndDialog](#) function.

A dialog box can contain up to 255 controls.

### Example

The following example uses the DialogBox function to create a modal dialog box:

```
DLGPROC dlgprc;
HWND hwndParent;

case IDM_ABOUT:
    dlgprc = (DLGPROC) MakeProcInstance(About, hinst);
    DialogBox(hinst, "AboutBox", hwndParent, dlgprc);
    FreeProcInstance((FARPROC) dlgprc);
    break;
```

### See Also

[DialogBoxIndirect](#), [DialogBoxIndirectParam](#), [DialogBoxParam](#), [DialogProc](#), [EndDialog](#), [GetDC](#), [MakeProcInstance](#), [WM\\_INITDIALOG](#)

## DialogBoxIndirect (2.x)

```
int DialogBoxIndirect(hinst, hglbDlgTemp, hwndOwner, dlgprc)
HINSTANCE hinst;          /* handle of application instance */
HGLOBAL hglbDlgTemp;
/* handle of memory with dialog box template
/
HWND hwndOwner;
/* handle of owner window
/
DLGPROC dlgprc;
/* instance address of dialog box procedure
/
```

The DialogBoxIndirect function creates a modal dialog box from a dialog box template in memory.

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst       | Identifies the instance of the module that will create the dialog box.                                                                                                                                                                                                                                                                                       |
| hglbDlgTemp | Identifies the global memory object that contains a dialog box template used to create the dialog box. This template is in the form of a DialogBoxHeader structure. For more information about this structure, see the <a href="#">Dialog Box Resource</a> topic.                                                                                            |
| hwndOwner   | Identifies the window that owns the dialog box.                                                                                                                                                                                                                                                                                                              |
| dlgprc      | Specifies the procedure-instance address of the dialog box procedure. The address must be created by using the <a href="#">MakeProcInstance</a> function, except when the function and dialog box procedure are used in a DLL. For more information about the dialog box procedure, see the description of the <a href="#">DialogProc</a> callback function. |

### Returns

The return value is the value of the nResult parameter specified in the [EndDialog](#) function that is used to terminate the dialog box. The system processes values returned by the [dialog box](#) procedure and does not return them to the application. The return value is -1 if the function cannot create the dialog box.

### Comments

The [CreateWindowEx](#) function is called to create the dialog box. The dialog box procedure then receives a [WM\\_SETFONT](#) message (if DS\_SETFONT style was specified) and a [WM\\_INITDIALOG](#) message, and then the dialog box is displayed.

The DialogBoxIndirect function does not return control until the dialog box procedure terminates the modal dialog box by calling the [EndDialog](#) function.

A dialog box can contain up to 255 controls.

### Example

The following example uses the DialogBoxIndirect function to create a dialog box from a dialog box template in memory:

```
#define TEMPLATE_SIZE 100
HGLOBAL hglbDlgTemp;
DLGPROC dlgprc;
int result;
HWND hwndParent;

/* Allocate a global memory object for the dialog box template. */
hglbDlgTemp = GlobalAlloc(GHND, TEMPLATE_SIZE);

.
. /* Build a DLGTEMPLATE structure in the memory object. */
.

dlgprc = (DLGPROC) MakeProcInstance(DialogProc, hinst);
result = DialogBoxIndirect(hinst, hglbDlgTemp, hwndParent, dlgprc);
```

See Also

DialogBox, DialogBoxIndirectParam, DialogBoxParam, DialogProc, EndDialog, MakeProcInstance,  
WM\_INITDIALOG

### DialogBoxIndirectParam (3.0)

```
int DialogBoxIndirectParam(hinst, hglbDlgTemp, hwndOwner, dlgproc, lParamInit)
HINSTANCE hinst;          /* handle of application instance */
HGLOBAL hglbDlgTemp;
/* handle of memory with dialog box template
/
HWND hwndOwner;
/* handle of owner window
/
DLGPROC dlgproc;
/* instance address of dialog box procedure
/
LPARAM lParamInit;
/* initialization value
/
```

The `DialogBoxIndirectParam` function creates a modal dialog box from a dialog box template in memory. Before displaying the dialog box, the function passes an application-defined value to the dialog box procedure as the `lParam` parameter of the `WM_INITDIALOG` message. An application can use this value to initialize dialog box controls.

| Parameter                | Description                                                                                                                                                                                                                                                                                                                                            |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>hinst</code>       | Identifies the instance of the module that will create the dialog box.                                                                                                                                                                                                                                                                                 |
| <code>hglbDlgTemp</code> | Identifies the global memory object that contains a dialog box template used to create the dialog box. This template is in the form of a <code>DialogBoxHeader</code> structure. For more information about this structure, see the <a href="#">Dialog Box Resource</a> topic.                                                                         |
| <code>hwndOwner</code>   | Identifies the window that owns the dialog box.                                                                                                                                                                                                                                                                                                        |
| <code>dlgproc</code>     | Specifies the procedure-instance address of the dialog box procedure. The address must be created by using the <code>MakeProcInstance</code> function, except when the function and dialog box procedure are used in a DLL. For more information about the dialog box procedure, see the description of the <code>DialogProc</code> callback function. |
| <code>lParamInit</code>  | Specifies a 32-bit value that <code>DialogBoxIndirectParam</code> passes to the dialog box when the <code>WM_INITDIALOG</code> message is being processed.                                                                                                                                                                                             |

#### Returns

The return value is the value of the `nResult` parameter specified in the `EndDialog` function that is used to terminate the dialog box. The system processes values returned by the dialog box procedure and does not return them to the application. The return value is -1 if the function cannot create the dialog box.

#### Comments

The `CreateWindowEx` function is called to create the dialog box. The dialog box procedure then receives a `WM_SETFONT` message (if `DS_SETFONT` style was specified) and a `WM_INITDIALOG` message, and then the dialog box is displayed.

The `DialogBoxIndirectParam` function does not return control until the dialog box procedure terminates the modal dialog box by calling the `EndDialog` function.

A dialog box can contain up to 255 controls.

#### Example

The following example uses the `DialogBoxIndirectParam` function to create a modal dialog box from a dialog box template in memory. The example uses the `lParamInit` parameter to send two initialization parameters (`wInitParm1` and `wInitParm2`) to the dialog box procedure when the `WM_INITDIALOG` message is being processed.

```
#define TEMPLATE_SIZE 100
HGLOBAL hglbDlgTemp;
DLGPROC dlgproc;
int result;
HWND hwndParent;
WORD wInitParm1, wInitParm2;

/* Allocate a global memory object for the dialog box template. */
```

```
hglbDlgTemp = GlobalAlloc(GHND, TEMPLATE_SIZE);
```

```
.  
. /* Build a DLGTEMPLATE structure in the memory object. */  
.
```

```
dlgprc = (DLGPROC) MakeProcInstance(DialogProc, hinst);  
result = DialogBoxIndirectParam(hinst, hglbDlgTemp, hwndParent,  
    dlgprc, (LPARAM) MAKELONG(wInitParm1, wInitParm2));
```

See Also

DialogBox, DialogBoxIndirect, DialogBoxParam, DialogProc, EndDialog, MakeProcInstance,  
WM\_INITDIALOG



## DialogBoxParam (3.0)

```
int DialogBoxParam(hinst, lpszDlgTemp, hwndOwner, dlgprc, lParamInit)
HINSTANCE hinst;          /* handle of application instance */
LPCSTR lpszDlgTemp;
/* address of dialog box template name
/
HWND hwndOwner;
/* handle of owner window
/
DLGPROC dlgprc;
/* instance address of dialog box procedure
/
LPARAM lParamInit;
/* initialization value
/
```

The DialogBoxParam function creates a modal dialog box from a dialog box template resource. Before displaying the dialog box, the function passes an application-specified value to the dialog box procedure as the lParam parameter of the WM\_INITDIALOG message. An application can use this value to initialize dialog box controls.

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst       | Identifies an instance of the module whose executable file contains the dialog box template.                                                                                                                                                                                                                                               |
| lpszDlgTemp | Points to a null-terminated string that names the dialog box template.                                                                                                                                                                                                                                                                     |
| hwndOwner   | Identifies the window that owns the dialog box.                                                                                                                                                                                                                                                                                            |
| dlgprc      | Specifies the procedure-instance address of the dialog box procedure. The address must be created by using the <u>MakeProcInstance</u> function, except when the function and dialog box procedure are used in a DLL. For more information about the dialog box procedure, see the description of the <u>DialogProc</u> callback function. |
| lParamInit  | Specifies a 32-bit value that DialogBoxParam passes to the dialog box procedure when creating the dialog box.                                                                                                                                                                                                                              |

### Returns

The return value specifies the value of the nResult parameter specified in the EndDialog function that is used to terminate the dialog box. The system processes values returned by the dialog box procedure and does not return them to the application. The return value is -1 if the function cannot create the dialog box.

### Comments

The CreateWindowEx function is called to create the dialog box. The dialog box procedure then receives a WM\_SETFONT message (if DS\_SETFONT style was specified) and a WM\_INITDIALOG message, and then the dialog box is displayed.

The DialogBoxParam function does not return control until the dialog box procedure terminates the modal dialog box by calling the EndDialog function.

A dialog box can contain up to 255 controls.

### Example

The following example uses the DialogBoxParam function to create a modal dialog box. The function passes the dialog box a pointer to a string when the WM\_INITDIALOG message is being processed.

```
DLGPROC dlgprc;
HWND hwndParent;
PSTR pszFileName;
int result;

case IDM_OPEN:

    dlgprc = (DLGPROC) MakeProcInstance(FileOpenProc, hinst);
    result = DialogBoxParam(hinst, "FileOpenBox", hwndParent,
        dlgprc, MAKELPARAM(pszFileName, 0));
    FreeProcInstance((FARPROC) dlgprc);
    break;
```

See Also

DialogBox, DialogBoxIndirect, DialogBoxIndirectParam, DialogProc, EndDialog, MakeProcInstance,  
WM\_INITDIALOG

## DispatchMessage (2.x)

```
LONG DispatchMessage(lpmsg)
const MSG FAR* lpmsg;          /* address of structure with message */
```

The DispatchMessage function dispatches a message to a window. It is typically used to dispatch a message retrieved by the [GetMessage](#) function.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpmsg     | Points to an <a href="#">MSG</a> structure that contains the message.<br>The <a href="#">MSG</a> structure must contain valid message values. If the lpmsg parameter points to a <a href="#">WM_TIMER</a> message and the lParam parameter of the <a href="#">WM_TIMER</a> message is not <a href="#">NULL</a> , then lParam points to a function that is called instead of the window procedure. |

### Returns

The return value specifies the value returned by the window procedure. Although its meaning depends on the message being dispatched, generally the return value is ignored.

### Example

The following example shows a typical use of the DispatchMessage function in an application's main message loop:

```
MSG msg;
HWND hwnd;
HWND hwndDlgModeless;
HANDLE hacc1;

while (GetMessage(&msg, NULL, 0, 0)) {
    if ((hwndDlgModeless == NULL ||
        !IsDialogMessage(hwndDlgModeless, &msg)) &&
        !TranslateAccelerator(hwnd, hacc1, &msg)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
}
```

### See Also

[GetMessage](#), [PeekMessage](#), [PostAppMessage](#), [PostMessage](#), [TranslateMessage](#), [MSG](#), [WM\\_TIMER](#)

## DlgDirList (2.x)

```
int DlgDirList(hwndDlg, lpszPath, idListBox, idStaticPath, uFileType)
HWND hwndDlg; /* handle of dialog box with list box */
LPSTR lpszPath; /* address of
path or filename string
/
int idListBox;
* identifier of list box
/
int idStaticPath;
* identifier of static control
/
UINT uFileType;
* file attributes to display
/
```

The DlgDirList function fills a list box with a file or directory listing. It fills the list box with the names of all files matching the specified path or filename.

| Parameter     | Description                                                                                                                                                                                                                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwndDlg       | Identifies the dialog box that contains the list box.                                                                                                                                                                                                                                                           |
| lpszPath      | Points to a null-terminated string that contains the path or filename. DlgDirList modifies this string, which should be long enough to contain the modifications. For more information, see the following Comments section.                                                                                     |
| idListBox     | Specifies the identifier of a list box. If this parameter is zero, DlgDirList assumes that no list box exists and does not attempt to fill one.                                                                                                                                                                 |
| idStaticPath  | Specifies the identifier of the static control used for displaying the current drive and directory. If this parameter is zero, DlgDirList assumes that no such control is present.                                                                                                                              |
| uFileType     | Specifies the attributes of the filenames to be displayed. This parameter can be a combination of the following values:                                                                                                                                                                                         |
| Value         | Meaning                                                                                                                                                                                                                                                                                                         |
| DDL_READWRITE | Read-write data files with no additional attributes.                                                                                                                                                                                                                                                            |
| DDL_READONLY  | Read-only files.                                                                                                                                                                                                                                                                                                |
| DDL_HIDDEN    | Hidden files.                                                                                                                                                                                                                                                                                                   |
| DDL_SYSTEM    | System files.                                                                                                                                                                                                                                                                                                   |
| DDL_DIRECTORY | Directories.                                                                                                                                                                                                                                                                                                    |
| DDL_ARCHIVE   | Archives.                                                                                                                                                                                                                                                                                                       |
| DDL_POSTMSGS  | <u>LB_DIR</u> flag. If the <u>LB_DIR</u> flag is set, Windows places the messages generated by DlgDirList in the application's queue; otherwise, they are sent directly to the dialog box procedure.                                                                                                            |
| DDL_DRIVES    | Drives. If the <u>DDL_DRIVES</u> flag is set, the <u>DDL_EXCLUSIVE</u> flag is set automatically. Therefore, to create a directory listing that includes drives and files, the developer must call DlgDirList twice: once with the <u>DDL_DRIVES</u> flag set and once with the flags for the rest of the list. |
| DDL_EXCLUSIVE | Exclusive bit. If the exclusive bit is set, only files of the specified type are listed; otherwise, files of the specified type are listed in addition to normal files.                                                                                                                                         |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

If you specify a zero-length string for the lpszPath parameter or if you specify only a directory name but do not include any filename, the string will be changed to \*.\*.

The DlgDirList function shows directories enclosed in brackets ([ ]) and shows drives in the form [-x-], where x is the drive letter.

The lpszPath parameter has the following form:

[drive:][[\\]directory[\\directory]...\\][filename]

In this example, drive is a drive letter, directory is a valid MS-DOS directory name, and filename is a valid MS-DOS filename that must contain at least one wildcard. The wildcards are a question mark (?), meaning match any character, and an asterisk (\*), meaning match any number of characters.

If the lpszPath parameter includes a drive or directory name, or both, the current drive and directory are changed to the specified drive and directory before the list box is filled. The static control identified by the idStaticPath parameter is also updated with the new drive or directory name, or both.

After the list box is filled, lpszPath is updated by removing the drive or directory portion, or both, of the path and filename.

DlgDirList sends LB\_RESETCONTENT and LB\_DIR messages to the list box.

See Also

DlgDirListComboBox, DlgDirSelect, DlgDirSelectComboBox, LB\_DIR, LB\_RESETCONTENT

### DlgDirListComboBox (3.0)

```
int DlgDirListComboBox(hwndDlg, lpszPath, idComboBox, idStaticPath, uFileType)
HWND hwndDlg;          /* handle of dialog box with combo box */
LPSTR lpszPath;        /* address of
path or filename string
/
int idComboBox;
* identifier of combo box
/
int idStaticPath;
* identifier of static control
/
UINT uFileType;
* file attributes to display
/
```

The DlgDirListComboBox function fills the list box of a combo box with a file or directory listing. It fills the list box with the names of all files matching the specified path and filename.

| Parameter            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|----------------------|------------------------------------------------------|---------------------|------------------|-------------------|---------------|-------------------|---------------|----------------------|--------------|--------------------|-----------|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwndDlg              | Identifies the dialog box that contains the combo box.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
| lpszPath             | Points to a null-terminated string that contains the path and filename. For more information, see the following Comments section.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
| idComboBox           | Specifies the identifier of a combo box in a dialog box. If this parameter is zero, DlgDirListComboBox assumes that no combo box exists and does not attempt to fill one.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
| idStaticPath         | Specifies the identifier of the static control used for displaying the current drive and directory. If this parameter is zero, DlgDirListComboBox assumes that no such control is present.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
| uFileType            | Specifies the attributes of the filenames to be displayed. This parameter can be a combination of the following values:<br><table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>DDL_READWRITE</u></td><td>Read-write data files with no additional attributes.</td></tr><tr><td><u>DDL_READONLY</u></td><td>Read-only files.</td></tr><tr><td><u>DDL_HIDDEN</u></td><td>Hidden files.</td></tr><tr><td><u>DDL_SYSTEM</u></td><td>System files.</td></tr><tr><td><u>DDL_DIRECTORY</u></td><td>Directories.</td></tr><tr><td><u>DDL_ARCHIVE</u></td><td>Archives.</td></tr><tr><td><u>DDL_POSTMSGS</u></td><td><u>CB_DIR</u> flag. If the <u>CB_DIR</u> flag is set, Windows places the messages generated by DlgDirListComboBox in the application's queue; otherwise, they are sent directly to the dialog box procedure.</td></tr><tr><td><u>DDL_DRIVES</u></td><td>Drives. If the <u>DDL_DRIVES</u> flag is set, the <u>DDL_EXCLUSIVE</u> flag is set automatically. Therefore, to create a directory listing that includes drives and files, the developer must call DlgDirListComboBox twice: once with the <u>DDL_DRIVES</u> flag set and once with the flags for the rest of the list.</td></tr><tr><td><u>DDL_EXCLUSIVE</u></td><td>Exclusive bit. If the exclusive bit is set, only files of the specified type are listed; otherwise, files of the specified type are listed in addition to normal files.</td></tr></tbody></table> | Value | Meaning | <u>DDL_READWRITE</u> | Read-write data files with no additional attributes. | <u>DDL_READONLY</u> | Read-only files. | <u>DDL_HIDDEN</u> | Hidden files. | <u>DDL_SYSTEM</u> | System files. | <u>DDL_DIRECTORY</u> | Directories. | <u>DDL_ARCHIVE</u> | Archives. | <u>DDL_POSTMSGS</u> | <u>CB_DIR</u> flag. If the <u>CB_DIR</u> flag is set, Windows places the messages generated by DlgDirListComboBox in the application's queue; otherwise, they are sent directly to the dialog box procedure. | <u>DDL_DRIVES</u> | Drives. If the <u>DDL_DRIVES</u> flag is set, the <u>DDL_EXCLUSIVE</u> flag is set automatically. Therefore, to create a directory listing that includes drives and files, the developer must call DlgDirListComboBox twice: once with the <u>DDL_DRIVES</u> flag set and once with the flags for the rest of the list. | <u>DDL_EXCLUSIVE</u> | Exclusive bit. If the exclusive bit is set, only files of the specified type are listed; otherwise, files of the specified type are listed in addition to normal files. |
| Value                | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
| <u>DDL_READWRITE</u> | Read-write data files with no additional attributes.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
| <u>DDL_READONLY</u>  | Read-only files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
| <u>DDL_HIDDEN</u>    | Hidden files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
| <u>DDL_SYSTEM</u>    | System files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
| <u>DDL_DIRECTORY</u> | Directories.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
| <u>DDL_ARCHIVE</u>   | Archives.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
| <u>DDL_POSTMSGS</u>  | <u>CB_DIR</u> flag. If the <u>CB_DIR</u> flag is set, Windows places the messages generated by DlgDirListComboBox in the application's queue; otherwise, they are sent directly to the dialog box procedure.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
| <u>DDL_DRIVES</u>    | Drives. If the <u>DDL_DRIVES</u> flag is set, the <u>DDL_EXCLUSIVE</u> flag is set automatically. Therefore, to create a directory listing that includes drives and files, the developer must call DlgDirListComboBox twice: once with the <u>DDL_DRIVES</u> flag set and once with the flags for the rest of the list.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |
| <u>DDL_EXCLUSIVE</u> | Exclusive bit. If the exclusive bit is set, only files of the specified type are listed; otherwise, files of the specified type are listed in addition to normal files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                      |                                                      |                     |                  |                   |               |                   |               |                      |              |                    |           |                     |                                                                                                                                                                                                              |                   |                                                                                                                                                                                                                                                                                                                         |                      |                                                                                                                                                                         |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The DlgDirListComboBox function shows directories enclosed in brackets ( [ ] ) and shows drives in the form [-x-], where x is the drive letter.

The lpszPath parameter has the following form:

[drive:][[\\]directory[\\directory]...\\][filename]

In this example, drive is a drive letter, directory is a valid MS-DOS directory name, and filename is a valid MS-DOS filename that must contain at least one wildcard. The wildcards are a question mark (?), meaning match any character, and an asterisk (\*), meaning match any number of characters.

If the lpszPath parameter includes a drive or directory name, or both, the current drive and directory are changed to the specified drive and directory before the list box is filled. The static control identified by the idStaticPath parameter is also updated with the new drive or directory name, or both.

After the list box of the combo box is filled, lpszPath is updated by removing the drive or directory portion, or both, of the path and filename.

DlgDirListComboBox sends CB\_RESETCONTENT and CB\_DIR messages to the combo box.

See Also

DlgDirList, DlgDirSelect, DlgDirSelectComboBox, CB\_DIR, CB\_RESETCONTENT

DDL\_READWRITE 0x0000

Read-write data files with no additional attributes.



DDL\_READWRITE 0x0000

DDL\_READONLY 0x0001

Read-only files.

DDL\_READONLY 0x0001

DDL\_HIDDEN 0x0002

Hidden files.

DDL\_HIDDEN 0x0002

DDL\_SYSTEM 0x0004  
System files.

DDL\_SYSTEM 0x0004

DDL\_DIRECTORY 0x0010

Directories.



DDL\_DIRECTORY 0x0010

DDL\_ARCHIVE 0x0020  
Archives.

DDL\_ARCHIVE 0x0020

DDL\_POSTMSG 0x2000

CB\_DIR flag. If the CB\_DIR flag is set, Windows places the messages generated by DlgDirListComboBox in the application's queue; otherwise, they are sent directly to the dialog box procedure.

DDL\_POSTMSG 0x2000

DDL\_DRIVES 0x4000

Drives. If the DDL\_DRIVES flag is set, the DDL\_EXCLUSIVE flag is set automatically. Therefore, to create a directory listing that includes drives and files, the developer must call DlgDirListComboBox twice: once with the DDL\_DRIVES flag set and once with the flags for the rest of the list.

DDL\_DRIVES 0x4000

DDL\_EXCLUSIVE 0x8000

Exclusive bit. If the exclusive bit is set, only files of the specified type are listed; otherwise, files of the specified type are listed in addition to normal files.



DDL\_EXCLUSIVE 0x8000

## DlgDirSelect (2.x)

```
BOOL DlgDirSelect(hwndDlg, lpszPath, idListBox)
HWND hwndDlg;      /* handle of dialog box with list box      */
LPSTR lpszPath;    /* address of buffer for path or filename string          */
/*
/
int idListBox;
/* identifier of list box
/
```

The DlgDirSelect function retrieves the current selection from a list box. It assumes that the list box has been filled by the [DlgDirList](#) function and that the selection is a drive letter, a file, or a directory name.

| Parameter | Description                                                                                               |
|-----------|-----------------------------------------------------------------------------------------------------------|
| hwndDlg   | Identifies the dialog box that contains the list box.                                                     |
| lpszPath  | Points to a buffer that will receive the selected path or filename. This buffer should be 128 bytes long. |
| idListBox | Specifies the integer identifier of a list box in the dialog box.                                         |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

If the current selection is a directory name or drive letter, DlgDirSelect removes the enclosing brackets (and hyphens, for drive letters) so that the name or letter is ready to be inserted into a new path or filename. If there is no selection, the contents of the buffer pointed to by the lpszPath parameter do not change.

The DlgDirSelect function does not allow more than one filename to be returned from a list box.

The list box must not be a multiple-selection list box. If it is, this function will not return a zero value and lpszPath will remain unchanged.

DlgDirSelect sends [LB\\_GETCURSEL](#) and [LB\\_GETTEXT](#) messages to the list box.

### See Also

[DlgDirList](#), [DlgDirListComboBox](#), [DlgDirSelectComboBox](#), [DlgDirSelectEx](#), [LB\\_GETCURSEL](#), [LB\\_GETTEXT](#)

## DlgDirSelectEx (2.x)

```
BOOL DlgDirSelectEx(HWND Dlg, LPSTR lpszPath, int cbPath, int idListBox)
/* handle of dialog box with list box */
/* address of
buffer for path string
/
int cbPath;
* number of bytes in path string
/
int idListBox;
* identifier of list box
/
```

The DlgDirSelectEx function retrieves the current selection from a list box. The specified list box should have been filled by the [DlgDirList](#) function, and the selection should be a drive letter, a file, or a directory name.

| Parameter | Description                                                                                                                             |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| hwndDlg   | Identifies the dialog box that contains the list box.                                                                                   |
| lpszPath  | Points to a buffer that receives the selected path or filename.                                                                         |
| cbPath    | Specifies the length, in bytes, of the path or filename pointed to by the lpszPath parameter. This value should not be larger than 128. |
| idListBox | Specifies the integer identifier of a list box in the dialog box.                                                                       |

### Returns

The return value is nonzero if the current list box selection is a directory name. Otherwise, it is zero.

### Comments

If the current selection is a directory name or drive letter, DlgDirSelectEx removes the enclosing square brackets (and hyphens, for drive letters) so that the name or letter is ready to be inserted into a new path or filename. If there is no selection, the contents of buffer pointed to by the lpszPath parameter do not change.

The DlgDirSelectEx function does not allow more than one filename to be returned from a list box.

The list box must not be a multiple-selection list box. If it is, this function will not return a zero value and lpszPath will remain unchanged.

DlgDirSelectEx sends [LB\\_GETCURSEL](#) and [LB\\_GETTEXT](#) messages to the list box.

### See Also

[DlgDirList](#), [DlgDirListComboBox](#), [DlgDirSelect](#), [DlgDirSelectComboBox](#), [LB\\_GETCURSEL](#), [LB\\_GETTEXT](#)

■ **DlgDirSelectComboBox (3.0)**

```
BOOL DlgDirSelectComboBox(hwndDlg, lpszPath, idComboBox)
HWND hwndDlg;          /* handle of dialog box with list box */
LPSTR lpszPath;        /*
address of buffer for path or filename string
/
int idComboBox;
* identifier of combo box
/
```

The DlgDirSelectComboBox function retrieves the current selection from the list box of a combo box. It assumes that the list box has been filled by the DlgDirListComboBox function and that the selection is a drive letter, a file, or a directory name.

| Parameter  | Description                                                                                               |
|------------|-----------------------------------------------------------------------------------------------------------|
| hwndDlg    | Identifies the dialog box that contains the combo box.                                                    |
| lpszPath   | Points to a buffer that will receive the selected path or filename. This buffer should be 128 bytes long. |
| idComboBox | Specifies the integer identifier of the combo box in the dialog box.                                      |

**Returns**

The return value is nonzero if the function is successful. Otherwise, it is zero.

**Comments**

The DlgDirSelectComboBox function does not allow more than one selection to be returned from a combo box.

If the current selection is a directory name or drive letter, DlgDirSelectComboBox removes the enclosing brackets (and hyphens, for drive letters) so that the name or letter is ready to be inserted into a new path or filename. If there is no selection, the contents of buffer pointed to by the lpszPath parameter do not change.

DlgDirSelectComboBox sends CB\_GETCURSEL and CB\_GETLBTEXT messages to the combo box.

**See Also**

DlgDirList, DlgDirListComboBox, DlgDirSelect, DlgDirSelectComboBoxEx, DlgDirSelectEx, CB\_GETCURSEL, CB\_GETLBTEXT

## Windows 3.1 Changes

The `DlgDirSelectComboBox` function now works with combo boxes that have the `CBS_DROPDOWN` or `CBS_DROPDOWNLIST` style.

### DlgDirSelectComboBoxEx (3.0)

```
BOOL DlgDirSelectComboBoxEx(HWND hDlg, LPCTSTR lpszPath, int cbPath, int idComboBox)
/* handle of dialog box with list box */
/* address of
buffer for path string
/
int cbPath;
* number of bytes in path string
/
int idComboBox;
* identifier of combo box
/
```

The DlgDirSelectComboBoxEx function retrieves the current selection from the list box of a combo box. The list box should have been filled by the DlgDirListComboBox function, and the selection should be a drive letter, a file, or a directory name.

| Parameter  | Description                                                                                                                             |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| hWndDlg    | Identifies the dialog box that contains the combo box.                                                                                  |
| lpszPath   | Points to a buffer that receives the selected path or filename.                                                                         |
| cbPath     | Specifies the length, in bytes, of the path or filename pointed to by the lpszPath parameter. This value should not be larger than 128. |
| idComboBox | Specifies the integer identifier of the combo box in the dialog box.                                                                    |

#### Returns

The return value is nonzero if the current combo box selection is a directory name. Otherwise, it is zero.

#### Comments

The DlgDirSelectComboBoxEx function does not allow more than one filename to be returned from a combo box.

If the current selection is a directory name or drive letter, DlgDirSelectComboBoxEx removes the enclosing square brackets (and hyphens, for drive letters) so that the name or letter is ready to be inserted into a new path or filename. If there is no selection, the contents of buffer pointed to by the lpszPath parameter do not change.

DlgDirSelectComboBoxEx sends CB\_GETCURSEL and CB\_GETLBTEXT messages to the combo box.

#### See Also

DlgDirList, DlgDirListComboBox, DlgDirSelect, DlgDirSelectEx, DlgDirSelectComboBox, CB\_GETCURSEL, CB\_GETLBTEXT

## DrawFocusRect (3.0)

```
void DrawFocusRect(hdc, lprc)
HDC hdc; /* handle of device context */
const RECT FAR* lprc; /* address of structure with rectangle */
/
```

The DrawFocusRect function draws a rectangle in the style used to indicate that the rectangle has the focus.

| Parameter | Description                                                                               |
|-----------|-------------------------------------------------------------------------------------------|
| hdc       | Identifies the device context.                                                            |
| lprc      | Points to a <u>RECT</u> structure that contains the logical coordinates of the rectangle. |

### Returns

This function does not return a value.

### Comments

Because this is an XOR function, calling it a second time and specifying the same rectangle removes the rectangle from the screen.

The rectangle this function draws cannot be scrolled. To scroll an area containing a rectangle drawn by this function, call DrawFocusRect to remove the rectangle from the screen, scroll the area, and then call DrawFocusRect to draw the rectangle in the new position.

### See Also

FrameRect, RECT

## DrawIcon (2.x)

```
BOOL DrawIcon(hdc, x, y, hicon)
HDC hdc;          /* handle of device context */
int x;           /* x-coordinate of upper-left
corner */
int y;           /* y-coordinate of upper-left corner */
HICON hicon;
/* handle of icon to draw */
```

The DrawIcon function draws an icon on the given device. The DrawIcon function places the icon's upper-left corner at the specified location.

| Parameter | Description                                                              |
|-----------|--------------------------------------------------------------------------|
| hdc       | Identifies the device context for a window.                              |
| x         | Specifies the logical x-coordinate of the upper-left corner of the icon. |
| y         | Specifies the logical y-coordinate of the upper-left corner of the icon. |
| hicon     | Identifies the icon to be drawn.                                         |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The icon resource must have been loaded by using the LoadIcon function. The MM\_TEXT mapping mode must be selected before using this function.

### See Also

GetMapMode, LoadIcon, SetMapMode



## DrawMenuBar (2.x)

```
void DrawMenuBar(hwnd)  
HWND hwnd;    /* handle of window with menu bar to redraw */
```

The DrawMenuBar function redraws the menu bar of the given window. If a menu bar is changed after Windows has created the window, an application should call this function to draw the changed menu bar.

| Parameter | Description                                       |
|-----------|---------------------------------------------------|
| hwnd      | Identifies the window whose menu must be redrawn. |

### Returns

This function does not return a value.

## DrawText (2.x)

```
int DrawText(hdc, lpsz, cb, lprc, fuFormat)
HDC hdc; /* handle of device context */
LPCSTR lpsz; /* address of string to draw */
int cb; /* string length */
RECT FAR* lprc; /* address of structure with formatting dimensions */
UINT fuFormat; /* text-drawing flags */
```

The DrawText function draws formatted text into a given rectangle. It formats text by expanding tabs into appropriate spaces, aligning text to the left, right, or center of the rectangle, and breaking text into lines that fit within the rectangle.

The DrawText function uses the device context's selected font, text color, and background color to draw the text. Unless the `DT_NOCLIP` format is specified, DrawText clips the text so that the text does not appear outside the given rectangle. All formatting is assumed to have multiple lines unless the `DT_SINGLELINE` format is specified.

| Parameter                       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|------------------------|----------------------------------------------------------------------------------------------|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|----------------------------|----------------------------|----------------------------------------------------------------------------|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------|----------------------|-------------------|------------------------|------------------------------------------------------------------------------------------|--------------------------|-----------------------------------------------------------------------------------------|
| hdc                             | Identifies the device context. This cannot be a metafile device context.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
| lpsz                            | Points to the string to be drawn. If the cb parameter is -1, the string must be null-terminated.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
| cb                              | Specifies the number of bytes in the string. If this parameter is -1, then the lpsz parameter is assumed to be a long pointer to a null-terminated string and DrawText computes the character count automatically.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
| lprc                            | Points to a <code>RECT</code> structure that contains the logical coordinates of the upper-left and lower-right corners of the rectangle in which the text is to be formatted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
| fuFormat                        | Specifies an array of flags that determine how to draw the text. This parameter can be a combination of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
|                                 | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><code>DT_BOTTOM</code></td><td>Specifies bottom-aligned text. This value must be combined with <code>DT_SINGLELINE</code>.</td></tr><tr><td><code>DT_CALCRECT</code></td><td>Determines the width and height of the rectangle. If there are multiple lines of text, DrawText will use the width of the rectangle pointed to by the lprc parameter and extend the base of the rectangle to bound the last line of text. If there is only one line of text, DrawText will modify the right side of the rectangle so that it bounds the last character in the line. In either case, DrawText returns the height of the formatted text but does not draw the text.</td></tr><tr><td><code>DT_CENTER</code></td><td>Centers text horizontally.</td></tr><tr><td><code>DT_EXPANDTABS</code></td><td>Expands tab characters. The default number of characters per tab is eight.</td></tr><tr><td><code>DT_EXTERNALLEADING</code></td><td>Includes the font external leading in line height. Normally, external leading is not included in the height of a line of text.</td></tr><tr><td><code>DT_LEFT</code></td><td>Left-aligns text.</td></tr><tr><td><code>DT_NOCLIP</code></td><td>Draws without clipping. DrawText is somewhat faster when <code>DT_NOCLIP</code> is used.</td></tr><tr><td><code>DT_NOPREFIX</code></td><td>Turns off processing of prefix characters. Normally, DrawText interprets the mnemonic &amp;</td></tr></tbody></table> | Value | Meaning | <code>DT_BOTTOM</code> | Specifies bottom-aligned text. This value must be combined with <code>DT_SINGLELINE</code> . | <code>DT_CALCRECT</code> | Determines the width and height of the rectangle. If there are multiple lines of text, DrawText will use the width of the rectangle pointed to by the lprc parameter and extend the base of the rectangle to bound the last line of text. If there is only one line of text, DrawText will modify the right side of the rectangle so that it bounds the last character in the line. In either case, DrawText returns the height of the formatted text but does not draw the text. | <code>DT_CENTER</code> | Centers text horizontally. | <code>DT_EXPANDTABS</code> | Expands tab characters. The default number of characters per tab is eight. | <code>DT_EXTERNALLEADING</code> | Includes the font external leading in line height. Normally, external leading is not included in the height of a line of text. | <code>DT_LEFT</code> | Left-aligns text. | <code>DT_NOCLIP</code> | Draws without clipping. DrawText is somewhat faster when <code>DT_NOCLIP</code> is used. | <code>DT_NOPREFIX</code> | Turns off processing of prefix characters. Normally, DrawText interprets the mnemonic & |
| Value                           | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
| <code>DT_BOTTOM</code>          | Specifies bottom-aligned text. This value must be combined with <code>DT_SINGLELINE</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
| <code>DT_CALCRECT</code>        | Determines the width and height of the rectangle. If there are multiple lines of text, DrawText will use the width of the rectangle pointed to by the lprc parameter and extend the base of the rectangle to bound the last line of text. If there is only one line of text, DrawText will modify the right side of the rectangle so that it bounds the last character in the line. In either case, DrawText returns the height of the formatted text but does not draw the text.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
| <code>DT_CENTER</code>          | Centers text horizontally.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
| <code>DT_EXPANDTABS</code>      | Expands tab characters. The default number of characters per tab is eight.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
| <code>DT_EXTERNALLEADING</code> | Includes the font external leading in line height. Normally, external leading is not included in the height of a line of text.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
| <code>DT_LEFT</code>            | Left-aligns text.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
| <code>DT_NOCLIP</code>          | Draws without clipping. DrawText is somewhat faster when <code>DT_NOCLIP</code> is used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |
| <code>DT_NOPREFIX</code>        | Turns off processing of prefix characters. Normally, DrawText interprets the mnemonic &                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                        |                                                                                              |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                            |                            |                                                                            |                                 |                                                                                                                                |                      |                   |                        |                                                                                          |                          |                                                                                         |

|                      |                                                                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | as a directive to underscore the character that follows, and the mnemonic && as a directive to print a single &. By specifying <u>DT_NOPREFIX</u> , this processing is turned off.                                          |
| <u>DT_RIGHT</u>      | Right-aligns text.                                                                                                                                                                                                          |
| <u>DT_SINGLELINE</u> | Specifies single line only. Carriage returns and linefeeds do not break the line.                                                                                                                                           |
| <u>DT_TABSTOP</u>    | Sets tab stops. The high-order byte of the fuFormat parameter is the number of characters for each tab. The default number of characters per tab is eight.                                                                  |
| <u>DT_TOP</u>        | Specifies top-aligned text (single line only).                                                                                                                                                                              |
| <u>DT_VCENTER</u>    | Specifies vertically centered text (single line only).                                                                                                                                                                      |
| <u>DT_WORDBREAK</u>  | Specifies word breaking. Lines are automatically broken between words if a word would extend past the edge of the rectangle specified by the lprc parameter. A carriage return– linefeed sequence will also break the line. |

Note that the DT\_CALCRECT, DT\_EXTERNALLEADING, DT\_INTERNAL, DT\_NOCLIP, and DT\_NOPREFIX values cannot be used with the DT\_TABSTOP value.

#### Returns

The return value specifies the height of the text if the function is successful.

#### Comments

If the selected font is too large for the specified rectangle, the DrawText function does not attempt to substitute a smaller font.

If the DT\_CALCRECT flag is specified, the RECT structure pointed to by the lprc parameter will be updated to reflect the width and height needed to draw the text.

If the TA\_UPDATECP text-alignment flag has been set (see the SetTextAlign function), DrawText will display text starting at the current position, rather than at the left of the given rectangle. DrawText will not wrap text when the TA\_UPDATECP flag has been set (the DT\_WORDBREAK flag will have no effect).

The text color must be set by the SetTextColor function.

#### See Also

ExtTextOut, SetTextColor, TabbedTextOut, TextOut, RECT

DT\_BOTTOM 0x0008

Specifies bottom-aligned text. This value must be combined with DT\_SINGLELINE.

DT\_BOTTOM 0x0008

DT\_CALCRECT 0x0400

Determines the width and height of the rectangle. If there are multiple lines of text, DrawText will use the width of the rectangle pointed to by the lprc parameter and extend the base of the rectangle to bound the last line of text. If there is only one line of text, DrawText will modify the right side of the rectangle so that it bounds the last character in the line. In either case, DrawText returns the height of the formatted text but does not draw the text.

DT\_CALCRECT 0x0400

DT\_CENTER 0x0001  
Centers text horizontally.



DT\_CENTER 0x0001

DT\_EXPANDTABS 0x0040

Expands tab characters. The default number of characters per tab is eight.

DT\_EXPANDTABS 0x0040

DT\_EXTERNALLEADING 0x0200

Includes the font external leading in line height. Normally, external leading is not included in the height of a line of text.

DT\_EXTERNALLEADING 0x0200

DT\_LEFT 0x0000

Left-aligns text.

DT\_LEFT 0x0000

DT\_NOCLIP 0x0100

Draws without clipping. DrawText is somewhat faster when DT\_NOCLIP is used.



DT\_NOCLIP 0x0100

DT\_NOPREFIX 0x0800

Turns off processing of prefix characters. Normally, DrawText interprets the mnemonic & as a directive to underscore the character that follows, and the mnemonic && as a directive to print a single &. By specifying DT\_NOPREFIX, this processing is turned off.

DT\_NOPREFIX 0x0800

DT\_RIGHT 0x0002

Right-aligns text.

DT\_RIGHT 0x0002

DT\_SINGLELINE 0x0020

Specifies single line only. Carriage returns and linefeeds do not break the line.

DT\_SINGLELINE 0x0020

DT\_TABSTOP 0x0080

Sets tab stops. The high-order byte of the fuFormat parameter is the number of characters for each tab. The default number of characters per tab is eight.



DT\_TABSTOP 0x0080

DT\_TOP 0x0000

Specifies top-aligned text (single line only).

DT\_TOP 0x0000

DT\_VCENTER 0x0004

Specifies vertically centered text (single line only).

DT\_VCENTER 0x0004

DT\_WORDBREAK 0x0010

Specifies word breaking. Lines are automatically broken between words if a word would extend past the edge of the rectangle specified by the `lprc` parameter. A carriage return–linefeed sequence will also break the line.

DT\_WORDBREAK 0x0010

## EmptyClipboard (2.x)

BOOL EmptyClipboard(void)

The EmptyClipboard function empties the clipboard and frees handles to data in the clipboard. It then assigns ownership of the clipboard to the window that currently has the clipboard open.

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The clipboard must be open when the EmptyClipboard function is called.

### See Also

OpenClipboard, WM\_DESTROYCLIPBOARD



### EnableCommNotification (3.1)

```
BOOL EnableCommNotification(idComDev, hwnd, cbWriteNotify, cbOutQueue)
int idComDev;          /* communications-device identifier */
HWND hwnd;            /* handle of window receiving messages */
int cbWriteNotify;    /* number of bytes written before notification */
int cbOutQueue;       /* minimum number of bytes in output queue */
```

The EnableCommNotification function enables or disables WM\_COMMNOTIFY message posting to the given window.

| Parameter     | Description                                                                                                                                                                                                                                          |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| idComDev      | Specifies the communications device that is posting notification messages to the window identified by the hwnd parameter. The <u>OpenComm</u> function returns the value for the idComDev parameter.                                                 |
| hwnd          | Identifies the window whose <u>WM_COMMNOTIFY</u> message posting will be enabled or disabled. If this parameter is <u>NULL</u> , EnableCommNotification disables message posting to the current window.                                              |
| cbWriteNotify | Indicates the number of bytes the COM driver must write to the application's input queue before sending a notification message. The message signals the application to read information from the input queue.                                        |
| cbOutQueue    | Indicates the minimum number of bytes in the output queue. When the number of bytes in the output queue falls below this number, the COM driver sends the application a notification message, signaling it to write information to the output queue. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero, indicating an invalid COM port identifier, a port that is not open, or a function not supported by COMM.DRV.

#### Comments

If an application specifies -1 for the cbWriteNotify parameter, the WM\_COMMNOTIFY message is sent to the specified window for CN\_EVENT and CN\_TRANSMIT notifications but not for CN\_RECEIVE notifications. If -1 is specified for the cbOutQueue parameter, CN\_EVENT and CN\_RECEIVE notifications are sent but CN\_TRANSMIT notifications are not.

If a timeout occurs before as many bytes as specified by the cbWriteNotify parameter are written to the input queue, a WM\_COMMNOTIFY message is sent with the CN\_RECEIVE flag set. When this occurs, another message will not be sent until the number of bytes in the input queue falls below the number specified in the cbWriteNotify parameter. Similarly, a WM\_COMMNOTIFY message in which the CN\_RECEIVE flag is set is sent only when the output queue is larger than the number of bytes specified in the cbOutQueue parameter.

The Windows 3.0 version of COMM.DRV does not support this function.

#### See Also

WM\_COMMNOTIFY

## EnableHardwareInput (2.x)

```
BOOL EnableHardwareInput(fEnableInput)
BOOL fEnableInput;      /* for enabling or disabling queuing */
```

The EnableHardwareInput function enables or disables queuing of mouse and keyboard input.

| Parameter    | Description                                                                                                                                                                                |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fEnableInput | Specifies whether to enable or disable queuing of input. If this parameter is TRUE, keyboard and mouse input are queued. If the parameter is FALSE, keyboard and mouse input are disabled. |

### Returns

The return value is nonzero if queuing of input was previously enabled. Otherwise, it is zero.

### Comments

This function does not disable input from installable drivers, nor does it disable device drivers.

### See Also

[GetInputState](#)

## EnableMenuItem (2.x)

```
BOOL EnableMenuItem(hmenu, idEnableItem, uEnable)
HMENU hmenu;          /* handle of menu */
UINT idEnableItem;    /* menu-item identifier */
/
UINT uEnable;
* action flag
/
```

The EnableMenuItem function enables, disables, or grays (dims) a menu item.

| Parameter     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |              |                                                                           |               |                                                                                                                     |             |                                           |            |                                          |           |                                         |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------------|---------------------------------------------------------------------------|---------------|---------------------------------------------------------------------------------------------------------------------|-------------|-------------------------------------------|------------|------------------------------------------|-----------|-----------------------------------------|
| hmenu         | Identifies the menu.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |              |                                                                           |               |                                                                                                                     |             |                                           |            |                                          |           |                                         |
| idEnableItem  | Specifies the menu item to be enabled, disabled, or grayed. This parameter can specify pop-up menu items as well as standard menu items. The interpretation of this parameter depends on the value of the uEnable parameter.                                                                                                                                                                                                                                                                                                                                                                           |       |         |              |                                                                           |               |                                                                                                                     |             |                                           |            |                                          |           |                                         |
| uEnable       | Specifies the action to take. This parameter can be MF_DISABLED, MF_ENABLED, or MF_GRAYED, combined with MF_BYCOMMAND or MF_BYPOSITION. These values have the following meanings:                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |              |                                                                           |               |                                                                                                                     |             |                                           |            |                                          |           |                                         |
|               | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MF_BYCOMMAND</td><td>Specifies that the idEnableItem parameter gives the menu-item identifier.</td></tr><tr><td>MF_BYPOSITION</td><td>Specifies that the idEnableItem parameter gives the position of the menu item (the first item is at position zero).</td></tr><tr><td>MF_DISABLED</td><td>Specifies that the menu item is disabled.</td></tr><tr><td>MF_ENABLED</td><td>Specifies that the menu item is enabled.</td></tr><tr><td>MF_GRAYED</td><td>Specifies that the menu item is grayed.</td></tr></tbody></table> | Value | Meaning | MF_BYCOMMAND | Specifies that the idEnableItem parameter gives the menu-item identifier. | MF_BYPOSITION | Specifies that the idEnableItem parameter gives the position of the menu item (the first item is at position zero). | MF_DISABLED | Specifies that the menu item is disabled. | MF_ENABLED | Specifies that the menu item is enabled. | MF_GRAYED | Specifies that the menu item is grayed. |
| Value         | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |              |                                                                           |               |                                                                                                                     |             |                                           |            |                                          |           |                                         |
| MF_BYCOMMAND  | Specifies that the idEnableItem parameter gives the menu-item identifier.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |              |                                                                           |               |                                                                                                                     |             |                                           |            |                                          |           |                                         |
| MF_BYPOSITION | Specifies that the idEnableItem parameter gives the position of the menu item (the first item is at position zero).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |              |                                                                           |               |                                                                                                                     |             |                                           |            |                                          |           |                                         |
| MF_DISABLED   | Specifies that the menu item is disabled.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |              |                                                                           |               |                                                                                                                     |             |                                           |            |                                          |           |                                         |
| MF_ENABLED    | Specifies that the menu item is enabled.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |              |                                                                           |               |                                                                                                                     |             |                                           |            |                                          |           |                                         |
| MF_GRAYED     | Specifies that the menu item is grayed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |              |                                                                           |               |                                                                                                                     |             |                                           |            |                                          |           |                                         |

### Returns

The return value is 0 if the menu item was previously disabled, 1 if the menu item was previously enabled, and -1 if the menu item does not exist.

### Comments

To disable or enable input to a menu bar, see the WM\_SYSCOMMAND message.

The CreateMenu, InsertMenu, ModifyMenu, and LoadMenuIndirect functions can also set the state (enabled, disabled, or grayed) of a menu item.

Using the MF\_BYPOSITION value requires an application to specify the correct menu handle. If the menu handle of the menu bar is specified, a top-level menu item (an item in the menu bar) is affected. To set the state of an item in a pop-up or nested pop-up menu by position, an application must specify the handle of the pop-up menu.

When an application specifies the MF\_BYCOMMAND flag, Windows checks all pop-up menu items that are subordinate to the menu identified by the specified menu handle; therefore, unless duplicate menu items are present, specifying the menu handle of the menu bar is sufficient.

### See Also

CheckMenuItem, HiliteMenuItem, WM\_SYSCOMMAND

### EnableScrollBar (3.1)

```
BOOL EnableScrollBar(HWND hwnd, fnSBFlags, fuArrowFlags)
/* handle of window or scroll bar */
int fnSBFlags; /* scroll-bar type
flag
/
UINT fuArrowFlags;
* scroll-bar arrow flag
/
```

The EnableScrollBar function enables or disables one or both arrows of a scroll bar.

| Parameter               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                        |                                                                                                                                                       |                         |                                                                                                           |                         |                                                                                                                                         |                         |                                                                                                                                       |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|-----------------------------------------------------------------------------------------------------------|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| hwnd                    | Identifies a window or a scroll bar, depending on the value of the fnSBFlags parameter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                        |                                                                                                                                                       |                         |                                                                                                           |                         |                                                                                                                                         |                         |                                                                                                                                       |
| fnSBFlags               | Specifies the scroll bar type. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><b>SB_BOTH</b></td><td>Enables or disables the arrows of the horizontal and vertical scroll bars associated with the given window. The hwnd parameter identifies the window.</td></tr><tr><td><b>SB_CTL</b></td><td>Identifies the scroll bar as a scroll bar control. The hwnd parameter must identify a scroll bar control.</td></tr><tr><td><b>SB_HORZ</b></td><td>Enables or disables the arrows of the horizontal scroll bar associated with the given window. The hwnd parameter identifies the window.</td></tr><tr><td><b>SB_VERT</b></td><td>Enables or disables the arrows of the vertical scroll bar associated with the given window. The hwnd parameter identifies the window.</td></tr></tbody></table> | Value | Meaning | <b>SB_BOTH</b>         | Enables or disables the arrows of the horizontal and vertical scroll bars associated with the given window. The hwnd parameter identifies the window. | <b>SB_CTL</b>           | Identifies the scroll bar as a scroll bar control. The hwnd parameter must identify a scroll bar control. | <b>SB_HORZ</b>          | Enables or disables the arrows of the horizontal scroll bar associated with the given window. The hwnd parameter identifies the window. | <b>SB_VERT</b>          | Enables or disables the arrows of the vertical scroll bar associated with the given window. The hwnd parameter identifies the window. |
| Value                   | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                        |                                                                                                                                                       |                         |                                                                                                           |                         |                                                                                                                                         |                         |                                                                                                                                       |
| <b>SB_BOTH</b>          | Enables or disables the arrows of the horizontal and vertical scroll bars associated with the given window. The hwnd parameter identifies the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                        |                                                                                                                                                       |                         |                                                                                                           |                         |                                                                                                                                         |                         |                                                                                                                                       |
| <b>SB_CTL</b>           | Identifies the scroll bar as a scroll bar control. The hwnd parameter must identify a scroll bar control.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                        |                                                                                                                                                       |                         |                                                                                                           |                         |                                                                                                                                         |                         |                                                                                                                                       |
| <b>SB_HORZ</b>          | Enables or disables the arrows of the horizontal scroll bar associated with the given window. The hwnd parameter identifies the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                        |                                                                                                                                                       |                         |                                                                                                           |                         |                                                                                                                                         |                         |                                                                                                                                       |
| <b>SB_VERT</b>          | Enables or disables the arrows of the vertical scroll bar associated with the given window. The hwnd parameter identifies the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                        |                                                                                                                                                       |                         |                                                                                                           |                         |                                                                                                                                         |                         |                                                                                                                                       |
| fuArrowFlags            | Specifies whether the scroll bar arrows are enabled or disabled, and which arrows are enabled or disabled. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><b>ESB_ENABLE_BOTH</b></td><td>Enables both arrows of a scroll bar.</td></tr><tr><td><b>ESB_DISABLE_LTUP</b></td><td>Disables the left arrow of a horizontal scroll bar, or the up arrow of a vertical scroll bar.</td></tr><tr><td><b>ESB_DISABLE_RTDN</b></td><td>Disables the right arrow of a horizontal scroll bar, or the down arrow of a vertical scroll bar.</td></tr><tr><td><b>ESB_DISABLE_BOTH</b></td><td>Disables both arrows of a scroll bar.</td></tr></tbody></table>                                                                                                                                                     | Value | Meaning | <b>ESB_ENABLE_BOTH</b> | Enables both arrows of a scroll bar.                                                                                                                  | <b>ESB_DISABLE_LTUP</b> | Disables the left arrow of a horizontal scroll bar, or the up arrow of a vertical scroll bar.             | <b>ESB_DISABLE_RTDN</b> | Disables the right arrow of a horizontal scroll bar, or the down arrow of a vertical scroll bar.                                        | <b>ESB_DISABLE_BOTH</b> | Disables both arrows of a scroll bar.                                                                                                 |
| Value                   | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                        |                                                                                                                                                       |                         |                                                                                                           |                         |                                                                                                                                         |                         |                                                                                                                                       |
| <b>ESB_ENABLE_BOTH</b>  | Enables both arrows of a scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                        |                                                                                                                                                       |                         |                                                                                                           |                         |                                                                                                                                         |                         |                                                                                                                                       |
| <b>ESB_DISABLE_LTUP</b> | Disables the left arrow of a horizontal scroll bar, or the up arrow of a vertical scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                        |                                                                                                                                                       |                         |                                                                                                           |                         |                                                                                                                                         |                         |                                                                                                                                       |
| <b>ESB_DISABLE_RTDN</b> | Disables the right arrow of a horizontal scroll bar, or the down arrow of a vertical scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                        |                                                                                                                                                       |                         |                                                                                                           |                         |                                                                                                                                         |                         |                                                                                                                                       |
| <b>ESB_DISABLE_BOTH</b> | Disables both arrows of a scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                        |                                                                                                                                                       |                         |                                                                                                           |                         |                                                                                                                                         |                         |                                                                                                                                       |

#### Returns

The return value is nonzero if the arrows are enabled or disabled as specified. Otherwise, it is zero, indicating that the arrows are already in the requested state or that an error occurred.

#### Example

The following example enables an edit control's vertical scroll bar when the control receives the input focus, and disables the scroll bar when the control loses the focus:

```
case EN_SETFOCUS :
    EnableScrollBar(hwndMLEdit, SB_VERT, ESB_ENABLE_BOTH);
    break;

case EN_KILLFOCUS :
    EnableScrollBar(hwndMLEdit, SB_VERT, ESB_DISABLE_BOTH);
    break;
```

#### See Also

[ShowScrollBar](#)

SB\_BOTH 3

Enables or disables the arrows of the horizontal and vertical scroll bars associated with the given window. The hwnd parameter identifies the window.

SB\_BOTH 3

SB\_CTL 2

Identifies the scroll bar as a scroll bar control. The hwnd parameter must identify a scroll bar control.

SB\_CTL 2



SB\_HORZ 0

Enables or disables the arrows of the horizontal scroll bar associated with the given window. The hwnd parameter identifies the window.

SB\_HORZ 0

SB\_VERT 1

Enables or disables the arrows of the vertical scroll bar associated with the given window. The hwnd parameter identifies the window.

SB\_VERT 1

ESB\_ENABLE\_BOTH 0x0000

Enables both arrows of a scroll bar.

ESB\_ENABLE\_BOTH 0x0000

ESB\_DISABLE\_LTUP ESB\_DISABLE\_LEFT

Disables the left arrow of a horizontal scroll bar, or the up arrow of a vertical scroll bar.

ESB\_DISABLE\_LTUP ESB\_DISABLE\_LEFT



ESB\_DISABLE\_RTDN ESB\_DISABLE\_RIGHT

Disables the right arrow of a horizontal scroll bar, or the down arrow of a vertical scroll bar.

ESB\_DISABLE\_RTDN ESB\_DISABLE\_RIGHT

ESB\_DISABLE\_BOTH 0x0003

Disables both arrows of a scroll bar.

ESB\_DISABLE\_BOTH 0x0003

## EnableWindow (2.x)

```
BOOL EnableWindow(hwnd, fEnable)
HWND hwnd;          /* handle of window          */
BOOL fEnable;       /* flag for enabling or disabling input */
/
```

The EnableWindow function enables or disables mouse and keyboard input to the given window or control. When input is disabled, the window ignores input such as mouse clicks and key presses. When input is enabled, the window processes all input.

| Parameter | Description                                                                                                                                             |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window to be enabled or disabled.                                                                                                        |
| fEnable   | Specifies whether to enable or disable the window. If this parameter is TRUE, the window is enabled. If the parameter is FALSE, the window is disabled. |

### Returns

The return value is nonzero if the window was previously disabled. Otherwise, the return value is zero.

### Comments

If the enabled state of the window is changing, a WM\_ENABLE message is sent before this function returns. If a window is already disabled, all its child windows are implicitly disabled, although they are not sent a WM\_ENABLE message.

A window must be enabled before it can be activated. For example, if an application is displaying a modeless dialog box and has disabled its main window, the application must enable the main window before destroying the dialog box. Otherwise, another window will receive the input focus and be activated. If a child window is disabled, it is ignored when Windows tries to determine which window should receive mouse messages.

By default, a window is enabled when it is created. An application can specify the WS\_DISABLED style in the CreateWindow or CreateWindowEx function to create a window that is initially disabled. After a window has been created, an application can use the EnableWindow function to enable or disable the window.

An application can use this function to enable or disable a control in a dialog box. A disabled control cannot receive the input focus, nor can a user access it.

### Example

The following example enables a Save push button in a dialog box, depending on whether a user-specified filename exists:

```
static char szFileName[128];

case WM_INITDIALOG:

    /* if a filename is specified, enable the Save push button. */

    EnableWindow(GetDlgItem(hdlg, IDOK),
        (szFileName[0] == '\\0' ? FALSE : TRUE));
    return TRUE;
```

### See Also

IsWindowEnabled, WM\_ENABLE

## EndDeferWindowPos (3.0)

```
BOOL EndDeferWindowPos(hwndp)
HDWP hwndp;    /* handle of internal structure */
```

The EndDeferWindowPos function simultaneously updates the position and size of one or more windows in a single screen-refresh cycle.

| Parameter | Description                                                                                                                                                                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwndp     | Identifies an internal structure that contains size and position information for one or more windows. This structure is returned by the <u>BeginDeferWindowPos</u> function or by the most recent call to the <u>DeferWindowPos</u> function. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

This function sends the WM\_WINDOWPOSCHANGING and WM\_WINDOWPOSCHANGED messages to each window identified in the internal structure.

### See Also

BeginDeferWindowPos, DeferWindowPos, WM\_WINDOWPOSCHANGED,  
WM\_WINDOWPOSCHANGING

## EndDialog (2.x)

```
void EndDialog(HWND hDlg, int nResult)
/* handle of dialog box */
/* value to return */
```

The EndDialog function hides a modal dialog box and causes the DialogBox function to return.

| Parameter | Description                                                              |
|-----------|--------------------------------------------------------------------------|
| hDlg      | Identifies the dialog box to be destroyed.                               |
| nResult   | Specifies the value that is returned to the caller of <u>DialogBox</u> . |

### Returns

This function does not return a value.

### Comments

The EndDialog function is required to complete processing of a modal dialog box created by the DialogBox function. An application calls EndDialog from within the dialog box procedure.

A dialog box procedure can call EndDialog at any time, even during the processing of the WM\_INITDIALOG message. If the function is called while WM\_INITDIALOG is being processed, the dialog box is hidden before it is shown and before the input focus is set.

EndDialog does not destroy the dialog box immediately. Instead, it sets a flag that directs Windows to destroy the dialog box when the DialogBox function returns.

### See Also

DialogBox, WM\_INITDIALOG

## EndPoint (2.x)

```
void EndPaint(hwnd, lpps)
HWND hwnd;                /* handle of window */
const PAINTSTRUCT FAR* lpps;
/* address of structure for paint data
/
```

The EndPaint function marks the end of painting in the given window. This function is required for each call to the BeginPaint function, but only after painting is complete.

| Parameter | Description                                                                                                                  |
|-----------|------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window that has been repainted.                                                                               |
| lpps      | Points to a <u>PAINTSTRUCT</u> structure that contains the painting information retrieved by the <u>BeginPaint</u> function. |

Returns  
This function does not return a value.

Comments  
If the caret was hidden by the BeginPaint function, the EndPaint function restores the caret to the screen.

See Also  
BeginPaint, PAINTSTRUCT



## EnumChildWindows (2.x)

```
BOOL EnumChildWindows(HWND hwndParent, WNDENUMPROC wndenumprc, LPARAM lParam)
/* handle of parent window */
/* address of callback function */
/* application-defined value */
```

The EnumChildWindows function enumerates the child windows that belong to the given parent window by passing the handle of each child window, in turn, to an application-defined callback function. EnumChildWindows continues until the last child window is enumerated or the callback function returns zero.

| Parameter  | Description                                                                                                                                                                                                                                                                                                      |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwndParent | Identifies the parent window whose child windows are to be enumerated.                                                                                                                                                                                                                                           |
| wndenumprc | Specifies the procedure-instance address of the application-supplied callback function. The address must have been created by using the <a href="#">MakeProcInstance</a> function. For more information about the callback function, see the <a href="#">description of the EnumChildProc</a> callback function. |
| lParam     | Specifies a 32-bit application-defined value to pass to the callback function.                                                                                                                                                                                                                                   |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

This function does not enumerate top-level windows that belong to the parent window.

If a child window has created child windows of its own, the function enumerates those windows as well.

A child window that is moved or repositioned in the Z-order during the enumeration process will be properly enumerated. The function will not enumerate a child window that is destroyed before it is enumerated or that is created during the enumeration process. These measures ensure that the EnumChildWindows function is reliable even when the application causes odd side effects, whereas an application that uses a [GetWindow](#) loop risks being caught in an infinite loop or referencing a handle to a window that has been destroyed.

### See Also

[EnumChildProc](#), [MakeProcInstance](#)

## EnumClipboardFormats (2.x)

```
UINT EnumClipboardFormats(uFormat)
UINT uFormat;          /* known clipboard format */
```

The EnumClipboardFormats function enumerates the formats found in a list of available formats that belong to the clipboard. Each call to this function specifies a known available format; the function returns the format that appears next in the list.

| Parameter | Description                                                                                             |
|-----------|---------------------------------------------------------------------------------------------------------|
| uFormat   | Specifies a known format. If this parameter is zero, the function returns the first format in the list. |

### Returns

The return value specifies the next known clipboard data format if the function is successful. It is zero if the uFormat parameter specifies the last format in the list of available formats, or if the clipboard is not open.

### Comments

Before it enumerates the formats by using the EnumClipboardFormats function, an application must open the clipboard by using the [OpenClipboard](#) function.

An application puts (or "donates") alternative formats for the same data into the clipboard in the same order that the enumerator uses when returning them to the pasting application. The pasting application should use the first format enumerated in the list that it can handle. This gives the donor application an opportunity to recommend formats that involve the least loss of data.

### See Also

[CountClipboardFormats](#), [GetClipboardFormatName](#), [GetPriorityClipboardFormat](#), [IsClipboardFormatAvailable](#), [OpenClipboard](#), [RegisterClipboardFormat](#)

## EnumProps (2.x)

```
int EnumProps(hwnd, prpenmproc)
HWND hwnd;          /* handle of window */
PROPENUMPROC prpenmproc;
/* address of callback function
/
```

The EnumProps function enumerates all entries in the property list of the given window. It enumerates the entries by passing them, one by one, to the specified callback function. EnumProps continues until the last entry is enumerated or the callback function returns zero.

| Parameter  | Description                                                                                                                                                                                                        |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd       | Identifies the window whose property list is enumerated.                                                                                                                                                           |
| prpenmproc | Specifies the procedure-instance address of the callback function. For more information, see the descriptions of the <a href="#">EnumPropFixedProc</a> and <a href="#">EnumPropMovableProc</a> callback functions. |

### Returns

The return value specifies the last value returned by the callback function. It is -1 if the function did not find a property to enumerate.

### Comments

The form of the callback function depends on whether the application or dynamic-link library (DLL) uses fixed or movable data segments. If the application or library uses fixed data segments (or if the library uses movable data segments that do not contain a stack), see the description of the [EnumPropFixedProc](#) callback function. If the application uses movable data segments (or if the library uses movable data segments that also contain a stack), see the description of the [EnumPropMovableProc](#) callback function.

An application's [EnumPropFixedProc](#) or [EnumPropMovableProc](#) callback function should not add new properties to a window. If the callback function deletes a window's properties, it should delete only the property currently being enumerated. The callback function should not delete other properties belonging to the window; if it does, the enumeration process terminates early.

The address passed in the prpenmproc parameter must be created by using the [MakeProcInstance](#) function.

### See Also

[EnumPropFixedProc](#), [EnumPropMovableProc](#), [GetProp](#), [MakeProcInstance](#), [RemoveProp](#), [SetProp](#)

## EnumTaskWindows (2.x)

```
BOOL EnumTaskWindows(htask, wndenmprc, lParam)
HTASK htask; /* handle of task */
WNDENUMPROC wndenmprc;
/* address of callback function */
LPARAM lParam;
/* application-defined value */
```

The EnumTaskWindows function enumerates all windows associated with a given task. (A task is any program that executes as an independent unit. All applications are executed as tasks, and each instance of an application is a task.) The function enumerates the windows by passing their handles, one by one, to the specified callback function. EnumTaskWindows continues until the last entry is enumerated or the callback function returns zero.

| Parameter | Description                                                                                                                                                   |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| htask     | Identifies the task. The task handle must be retrieved by a previous call to the <u>GetCurrentTask</u> function.                                              |
| wndenmprc | Specifies the procedure-instance address of the callback function. For more information, see the description of the <u>EnumTaskWndProc</u> callback function. |
| lParam    | Specifies a 32-bit application-defined value that is passed to the callback function along with each window handle.                                           |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

This function enumerates all top-level windows but does not enumerate child windows.

The EnumTaskWindows function is reliable even when the application causes odd side effects, whereas an application that uses a GetWindow loop risks being caught in an infinite loop or referencing a handle to a window that has been destroyed.

The address passed in the wndenmprc parameter must be created by using the MakeProcInstance function.

### See Also

EnumTaskWndProc, GetCurrentTask

## EnumWindows (2.x)

```
BOOL EnumWindows(WNDENUMPROC wndenumproc, LPARAM lParam)
/* address of callback function */
/* application-defined value */
```

The EnumWindows function enumerates all parent windows on the screen by passing the handle of each window, in turn, to an application-defined callback function. EnumWindows continues until the last parent window is enumerated or the callback function returns zero.

| Parameter   | Description                                                                                                                                                   |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wndenumproc | Specifies the procedure-instance address of the callback function. For more information, see the description of the <u>EnumWindowsProc</u> callback function. |
| lParam      | Specifies a 32-bit application-defined value that is passed to the callback function.                                                                         |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The EnumWindows function does not enumerate child windows.

EnumWindows is reliable even when the application causes odd side effects, whereas an application that uses a GetWindow loop risks being caught in an infinite loop or referencing a handle to a window that has been destroyed.

The address passed as the wndenumproc parameter must be created by using the MakeProcInstance function.

### See Also

EnumWindowsProc, MakeProcInstance

## EqualRect (2.x)

```
BOOL EqualRect(lprc1, lprc2)
const RECT FAR* lprc1;          /* address of structure with first rectangle */
const RECT FAR* lprc2;          /* address of structure with second rectangle */
/
```

The EqualRect function determines whether the two given rectangles are equal by comparing the coordinates of their upper-left and lower-right corners.

| Parameter | Description                                                                                      |
|-----------|--------------------------------------------------------------------------------------------------|
| lprc1     | Points to a <u>RECT</u> structure that contains the logical coordinates of the first rectangle.  |
| lprc2     | Points to a <u>RECT</u> structure that contains the logical coordinates of the second rectangle. |

### Returns

The return value is nonzero if the two rectangles are identical. Otherwise, it is zero.

### See Also

RECT

## EscapeCommFunction (2.x)

```
LONG EscapeCommFunction(idComDev, nFunction)
int idComDev;      /* identifies communications device */
int nFunction;     /* code of extended function */
function          /* code of extended function */
/
```

The EscapeCommFunction function directs the specified communications device to carry out an extended function.

| Parameter | Description                                                                                                                                                                      |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| idComDev  | Specifies the communications device that will carry out the extended function. The <u>OpenComm</u> function returns this value.                                                  |
| nFunction | Specifies the function code of the extended function. It can be one of the following values:                                                                                     |
| Value     | Meaning                                                                                                                                                                          |
| CLRDRTR   | Clears the DTR (data-terminal-ready) signal.                                                                                                                                     |
| CLRRTS    | Clears the RTS (request-to-send) signal.                                                                                                                                         |
| GETMAXCOM | Returns the maximum COM port identifier supported by the system. This value ranges from 0x00 to 0x7F, such that 0x00 corresponds to COM1, 0x01 to COM2, 0x02 to COM3, and so on. |
| GETMAXLPT | Returns the maximum LPT port identifier supported by the system. This value ranges from 0x80 to 0xFF, such that 0x80 corresponds to LPT1, 0x81 to LPT2, 0x82 to LPT3, and so on. |
| RESETDEV  | Resets the printer device if the idComDev parameter specifies an LPT port. No function is performed if idComDev specifies a COM port.                                            |
| SETDTR    | Sends the DTR (data-terminal-ready) signal.                                                                                                                                      |
| SETRTS    | Sends the RTS (request-to-send) signal.                                                                                                                                          |
| SETXOFF   | Causes transmission to act as if an XOFF character has been received.                                                                                                            |
| SETXON    | Causes transmission to act as if an XON character has been received.                                                                                                             |

### Returns

The return value is zero if the function is successful. Otherwise, it is less than zero.

## ExcludeUpdateRgn (2.x)

```
int ExcludeUpdateRgn(hdc, hwnd)
HDC hdc;          /* handle of device context */
HWND hwnd;       /* handle of window */
```

The ExcludeUpdateRgn function prevents drawing within invalid areas of a window by excluding an updated region in the window from a clipping region.

| Parameter | Description                                                        |
|-----------|--------------------------------------------------------------------|
| hdc       | Identifies the device context associated with the clipping region. |
| hwnd      | Identifies the window to be updated.                               |

### Returns

The return value is SIMPLEREGION (region has no overlapping borders), COMPLEXREGION (region has overlapping borders), or NULLREGION (region is empty), if the function is successful. Otherwise, the return value is ERROR (no region is created).

### See Also

[BeginPaint](#), [GetUpdateRect](#), [GetUpdateRgn](#), [UpdateWindow](#)



■ **ExitWindows (3.0)**

```

BOOL ExitWindows(dwReturnCode, reserved)
DWORD dwReturnCode;    /* return or restart code */
UINT reserved;        /* reserved; must be zero */
/

```

The ExitWindows function can restart Windows, terminate Windows and return control to MS-DOS, or terminate Windows and restart the system. Windows sends the WM\_QUERYENDSESSION message to notify all applications that a request has been made to restart or terminate Windows. If all applications "agree" to terminate, Windows sends the WM\_ENDSESSION message to all applications before terminating.

| Parameter                | Description                                                                                                                                                                                                                                                                                                                       |       |         |                        |                                                        |                          |                            |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|------------------------|--------------------------------------------------------|--------------------------|----------------------------|
| dwReturnCode             | Specifies whether Windows should restart, terminate and return control to MS-DOS, or terminate and restart the system. The high-order word of this parameter should be zero. The low-order word specifies the return value to be passed to MS-DOS when Windows terminates. The low-order word can be one of the following values: |       |         |                        |                                                        |                          |                            |
|                          | <table border="0"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td><u>EW_REBOOTSYSTEM</u></td> <td>Causes Windows to terminate and the system to restart.</td> </tr> <tr> <td><u>EW_RESTARTWINDOWS</u></td> <td>Causes Windows to restart.</td> </tr> </tbody> </table>                              | Value | Meaning | <u>EW_REBOOTSYSTEM</u> | Causes Windows to terminate and the system to restart. | <u>EW_RESTARTWINDOWS</u> | Causes Windows to restart. |
| Value                    | Meaning                                                                                                                                                                                                                                                                                                                           |       |         |                        |                                                        |                          |                            |
| <u>EW_REBOOTSYSTEM</u>   | Causes Windows to terminate and the system to restart.                                                                                                                                                                                                                                                                            |       |         |                        |                                                        |                          |                            |
| <u>EW_RESTARTWINDOWS</u> | Causes Windows to restart.                                                                                                                                                                                                                                                                                                        |       |         |                        |                                                        |                          |                            |
| reserved                 | Reserved; must be zero.                                                                                                                                                                                                                                                                                                           |       |         |                        |                                                        |                          |                            |

**Returns**

The return value is zero if one or more applications refuse to terminate. The function does not return a value if all applications agree to be terminated.

**See Also**

ExitWindowsExec, WM\_ENDSESSION, WM\_QUERYENDSESSION

### Windows 3.1 changes

An application can restart Windows by specifying a value of EW\_RESTARTWINDOWS as the dwReturnCode parameter of the ExitWindows function. This change is supported in Windows version 3.0 and 3.1.

The EW\_REBOOTSYSYSTEM flag can be specified as the low-order word of the dwReturnCode parameter. This flag causes Windows to terminate and the system to restart.

EW\_REBOOTSYSYSTEM 0x43

Causes Windows to terminate and the system to restart.

EW\_REBOOTSYSYSTEM 0x43

EW\_RESTARTWINDOWS 0x42

Causes Windows to restart.

EW\_RESTARTWINDOWS 0x42

## ExitWindowsExec (3.0)

BOOL ExitWindowsExec(lpszExe, lpszParams)  
LPCSTR lpszExe;  
LPCSTR lpszParams;

The ExitWindowsExec function terminates Windows, runs a specified MS-DOS application, and then restarts Windows.

| Parameter  | Description                                                                                                                                                                                                                                   |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszExe    | Points to a null-terminated string specifying the path and filename of the executable file for the system to run after Windows has been terminated. This string must not be longer than 128 bytes (including the null terminating character). |
| lpszParams | Points to a null-terminated string specifying any parameters for the executable file specified by the lpszExe parameter. This string must not be longer than 127 bytes (including the null terminating character). This value can be NULL.    |

### Returns

The return value is FALSE if the function fails. (The function could fail because of a memory-allocation error or if one of the applications in the system does not terminate.)

### Comments

The ExitWindowsExec function is typically used by installation programs to replace components of Windows which are active when Windows is running.

### See Also

[ExitWindows](#)

## FillRect (2.x)

```
int FillRect(hdc, lprc, hbr)
HDC hdc;                /* handle of device context */
const RECT FAR* lprc;   /* address of structure with rectangle */
HBRUSH hbr;            /* handle of brush */
```

The FillRect function fills a given rectangle by using the specified brush. The FillRect function fills the complete rectangle, including the left and top borders, but does not fill the right and bottom borders.

| Parameter | Description                                                                                            |
|-----------|--------------------------------------------------------------------------------------------------------|
| hdc       | Identifies the device context.                                                                         |
| lprc      | Points to a <u>RECT</u> structure that contains the logical coordinates of the rectangle to be filled. |
| hbr       | Identifies the brush used to fill the rectangle.                                                       |

### Returns

The return value is not used and has no meaning.

### Comments

The brush must be created by using either the CreateHatchBrush, CreatePatternBrush, or CreateSolidBrush function, or retrieved by using the GetStockObject function.

When filling the specified rectangle, the FillRect function does not include the rectangle's right and bottom sides. Graphics device interface (GDI) fills a rectangle up to, but not including, the right column and bottom row, regardless of the current mapping mode.

FillRect compares the values of the top, bottom, left, and right members of the specified RECT structure. If bottom is less than or equal to top, or if right is less than or equal to left, the function does not draw the rectangle.

### See Also

CreateHatchBrush, CreatePatternBrush, CreateSolidBrush, GetStockObject, InvertRect, RECT



## FindWindow (2.x)

```
HWND FindWindow(lpszClassName, lpszWindow)
LPCSTR lpszClassName;          /* address of class-name string */
LPCSTR lpszWindow;            /* address of window-name string */
```

The FindWindow function retrieves the handle of the window whose class name and window name match the specified strings. This function does not search child windows.

| Parameter     | Description                                                                                                                                |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| lpszClassName | Points to a null-terminated string that contains the window's class name. If this parameter is NULL, all class names match.                |
| lpszWindow    | Points to a null-terminated string that specifies the window name (the window's title). If this parameter is NULL, all window names match. |

### Returns

The return value is the handle of the window that has the specified class name and window name if the function is successful. Otherwise, it is NULL.

### Example

The following example searches for the main window of Windows Control Panel (CONTROL.EXE) and, if it does not find it, starts Control Panel:

```
if (FindWindow("CtlPanelClass", "Control Panel") == NULL)
    WinExec("control.exe", SW_SHOWNA);
```

### See Also

EnumWindows, GetWindow, WindowFromPoint

## FlashWindow (2.x)

```
BOOL FlashWindow(hwnd, fInvert)
HWND hwnd;      /* handle of window to flash */
BOOL fInvert;   /* invert flag */
```

The FlashWindow function flashes the given window once. Flashing a window means changing the appearance of its title bar as if the window were changing from inactive to active status or vice versa. (An inactive title bar changes to an active title bar or an active title bar changes to an inactive title bar.)

Typically, a window is flashed to inform the user that the window requires attention but that it does not currently have the input focus.

| Parameter | Description                                                                                                                                                                                                                                                  |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window to be flashed. The window can be either open or minimized.                                                                                                                                                                             |
| fInvert   | Specifies whether to flash the window or return it to its original state. If this parameter is TRUE, the window is flashed from one state to the other. If the parameter is FALSE, the window is returned to its original state (either active or inactive). |

### Returns

The return value is nonzero if the window was active before the call to the FlashWindow function. Otherwise, it is zero.

### Comments

The FlashWindow function flashes the window only once; for successive flashing, the application should create a system timer.

The fInvert parameter should be FALSE only when the window is receiving the input focus and will no longer be flashing; it should be TRUE on successive calls while waiting to get the input focus.

This function always returns nonzero for minimized windows. If the window is minimized, FlashWindow simply flashes the window's icon; fInvert is ignored for minimized windows.

### See Also

[MessageBeep](#)

## FlushComm (2.x)

```
int FlushComm(idComDev, fnQueue)
int idComDev;      /* communications-device identifier */
int fnQueue;      /* queue to flush */
```

The FlushComm function flushes all characters from the transmission or receiving queue of the specified communications device.

| Parameter | Description                                                                                                                                             |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| idComDev  | Specifies the communication device to be flushed. The <u>OpenComm</u> function returns this value.                                                      |
| fnQueue   | Specifies the queue to be flushed. If this parameter is zero, the transmission queue is flushed. If the parameter is 1, the receiving queue is flushed. |

### Returns

The return value is zero if the function is successful. It is less than zero if idComDev is not a valid device or if fnQueue is not a valid queue. The return value is positive if there is an error for the specified device. For a list of the possible error values, see the GetCommError function.

### See Also

GetCommError, OpenComm

## FrameRect (2.x)

```
int FrameRect(hdc, lprc, hbr)
HDC hdc; /* handle of device context */
const RECT FAR* lprc; /* address of structure with rectangle */
HBRUSH hbr; /* handle of brush */
```

The FrameRect function draws a border around a rectangle, using the specified brush. The width and height of the border are always one logical unit.

| Parameter | Description                                                                                                                                |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------|
| hdc       | Identifies the device context in which to draw the border.                                                                                 |
| lprc      | Points to a <u>RECT</u> structure that contains the logical coordinates of the upper-left and lower-right <u>corners</u> of the rectangle. |
| hbr       | Identifies the brush that will be used to draw the border.                                                                                 |

### Returns

The return value is not used and has no meaning.

### Comments

The border drawn by the FrameRect function is in the same position as a border drawn by the Rectangle function using the same coordinates (if Rectangle uses a pen that is one logical unit wide). The interior of the rectangle is not filled when an application calls FrameRect.

FrameRect compares the values of the top, bottom, left, and right members of the specified RECT structure. If bottom is less than or equal to top, or if right is less than or equal to left, FrameRect does not draw the rectangle.

### See Also

CreateHatchBrush, CreatePatternBrush, CreateSolidBrush, DrawFocusRect, RECT

## GetActiveWindow (2.x)

HWND GetActiveWindow(void)

The GetActiveWindow function retrieves the window handle of the active window. The active window is either the top-level window associated with the input focus or the window explicitly made active by the SetActiveWindow function.

### Returns

The return value is the handle of the active window or NULL if no window was active at the time of the call.

### See Also

GetCapture, GetFocus, GetLastActivePopup, SetActiveWindow

## GetAsyncKeyState (2.x)

```
int GetAsyncKeyState(vkey)
int vkey;    /* virtual-key code */
```

The `GetAsyncKeyState` function determines whether a key is up or down at the time the function is called and whether the key was pressed after a previous call to the `GetAsyncKeyState` function.

| Parameter         | Description                                      |
|-------------------|--------------------------------------------------|
| <code>vkey</code> | Specifies one of 256 possible virtual-key codes. |

### Returns

The return value specifies whether the key was pressed since the last call to the `GetAsyncKeyState` function and whether the key is currently up or down. If the most significant bit is set, the key is down, and if the least significant bit is set, the key was pressed after a preceding `GetAsyncKeyState` call.

### Comments

If `VK_LBUTTON` or `VK_RBUTTON` is specified in the `vkey` parameter, this function returns the state of the physical left or right mouse button regardless of whether the [SwapMouseButton](#) function has been used to reverse the meaning of the buttons.

### See Also

[GetKeyboardState](#), [GetKeyState](#), [SetKeyboardState](#), [SwapMouseButton](#)

## GetCapture (2.x)

HWND GetCapture(void)

The GetCapture function retrieves a handle of the window that has the mouse capture. Only one window has the mouse capture at any given time; this window receives mouse input whether or not the cursor is within its borders.

### Returns

The return value is a handle identifying the window that has the mouse capture if the function is successful. It is NULL if no window has the mouse capture.

### Comments

A window receives the mouse capture when its handle is passed as the hwnd parameter of the SetCapture function.

### See Also

SetCapture

## GetCaretBlinkTime (2.x)

UINT GetCaretBlinkTime(void)

The GetCaretBlinkTime function retrieves the caret blink rate. The blink rate is the elapsed time, in milliseconds, between flashes of the caret.

### Returns

The return value specifies the blink rate, in milliseconds, if the function is successful.

### See Also

[SetCaretBlinkTime](#)



## GetCaretPos (2.x)

```
void GetCaretPos(lppt)
POINT FAR* lppt;          /* address of structure to receive coordinates */
```

The GetCaretPos function retrieves the current position of the caret.

| Parameter | Description                                                                                              |
|-----------|----------------------------------------------------------------------------------------------------------|
| lppt      | Points to a <u>POINT</u> structure that receives the client coordinates of the caret's current position. |

### Returns

This function does not return a value.

### Comments

The caret position is always given in the client coordinates of the window that contains the caret.

### See Also

SetCaretPos, POINT

## GetClassInfo (3.0)

```
BOOL GetClassInfo(hinst, lpszClassName, lpwc)
HINSTANCE hinst;          /* handle of application instance */
LPCSTR lpszClassName;    /* address of class-name string */
WNDCLASS FAR* lpwc;      /* address of structure for class data */
```

The `GetClassInfo` function retrieves information about a window class. This function is used for creating subclasses of a given class.

| Parameter                  | Description                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>hinst</code>         | Identifies the instance of the application that created the class. To retrieve information about classes defined by Windows (such as buttons or list boxes), set this parameter to <code>NULL</code> .                                                                                                                                                                                                                  |
| <code>lpszClassName</code> | Points to a null-terminated string containing the class name. The class name is either an application-specified name as defined by the <code>RegisterClass</code> function or the name of a preregistered window class. If the high-order word of this parameter is <code>NULL</code> , the low-order word is assumed to be a value returned by the <code>MAKEINTRESOURCE</code> macro used when the class was created. |
| <code>lpwc</code>          | Points to a <code>WNDCLASS</code> structure that receives the information about the class.                                                                                                                                                                                                                                                                                                                              |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero, indicating the function did not find a matching class.

### Comments

The `GetClassInfo` function does not set the `lpszClassName` and `lpszMenuName` members of the `WNDCLASS` structure. The menu name is not stored internally and cannot be returned. The class name is already known, since it is passed to this function. `GetClassInfo` returns all other members with the values used when the class was registered.

### See Also

[GetClassLong](#), [GetClassName](#), [GetClassWord](#), [RegisterClass](#), [MAKEINTRESOURCE](#), [WNDCLASS](#)

## GetClassLong (2.x)

```
LONG GetClassLong(hwnd, offset)
HWND hwnd;    /* handle of window */
int offset;   /* offset of value to retrieve */
```

The `GetClassLong` function retrieves a 32-bit (long) value at the specified offset into the extra class memory for the window class to which the given window belongs. Extra class memory is reserved by specifying a nonzero value in the `cbClsExtra` member of the WNDCLASS structure used with the RegisterClass function.

| Parameter           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                     |                                                     |                    |                                                     |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---------------------|-----------------------------------------------------|--------------------|-----------------------------------------------------|
| <code>hwnd</code>   | Identifies the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                     |                                                     |                    |                                                     |
| <code>offset</code> | Specifies the zero-based byte offset of the value to be retrieved. Valid values are in the range zero through the number of bytes of class memory minus four (for example, if 12 or more bytes of extra class memory was specified, a value of 8 would be an index to the third 32-bit integer) or one of the following values:<br><table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>GCL_MENUNAME</u></td><td>Retrieves a 32-bit pointer to the menu-name string.</td></tr><tr><td><u>GCL_WNDPROC</u></td><td>Retrieves a 32-bit pointer to the window procedure.</td></tr></tbody></table> | Value | Meaning | <u>GCL_MENUNAME</u> | Retrieves a 32-bit pointer to the menu-name string. | <u>GCL_WNDPROC</u> | Retrieves a 32-bit pointer to the window procedure. |
| Value               | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                     |                                                     |                    |                                                     |
| <u>GCL_MENUNAME</u> | Retrieves a 32-bit pointer to the menu-name string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                     |                                                     |                    |                                                     |
| <u>GCL_WNDPROC</u>  | Retrieves a 32-bit pointer to the window procedure.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                     |                                                     |                    |                                                     |

### Returns

The return value is the specified 32-bit value in the extra class memory if the function is successful. Otherwise, it is zero, indicating the `hwnd` or `offset` parameter is invalid.

### Comments

To access any extra four-byte values allocated when the window-class structure was created, use a positive byte offset as the index specified by the `offset` parameter, starting at 0 for the first four-byte value in the extra space, 4 for the next four-byte value, and so on.

### See Also

GetClassInfo, GetClassName, GetClassWord, RegisterClass, SetClassLong, WNDCLASS

Windows 3.1 changes

Value

Meaning

GCL\_MENUNAME      Retrieves a 32-bit pointer to the menu-name string.

GCL\_MENUNAME (-8)

Retrieves a 32-bit pointer to the menu-name string.

GCL\_MENUNAME (-8)

GCL\_WNDPROC (-24)

Retrieves a 32-bit pointer to the window procedure.

GCL\_WNDPROC (-24)



## GetClassName (2.x)

```
int GetClassName(hwnd, lpszClassName, cchClassName)
HWND hwnd;          /* handle of window */
LPSTR lpszClassName; /* address of buffer for class name */
int cchClassName;  /* size of buffer */
```

The GetClassName function retrieves the class name of a window.

| Parameter     | Description                                                                                                                                       |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd          | Identifies the window.                                                                                                                            |
| lpszClassName | Points to a buffer that receives the null-terminated class name string.                                                                           |
| cchClassName  | Specifies the length of the buffer pointed to by the lpszClassName parameter. The class name string is truncated if it is longer than the buffer. |

### Returns

The return value is the length, in bytes, of the returned class name, not including the terminating null character. The return value is zero if the specified window handle is invalid.

## GetClassWord (2.x)

```
WORD GetClassWord(hwnd, offset)
HWND hwnd;      /* handle of window      */
int offset;     /* offset of value to retrieve */
```

The `GetClassWord` function retrieves a 16-bit (word) value at the specified offset into the extra class memory for the window class to which the given window belongs. Extra class memory is reserved by specifying a nonzero value in the `cbClsExtra` member of the WNDCLASS structure used with the RegisterClass function.

| Parameter                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                       |                                                                                                                                                     |                       |                                                                                                                                                      |                          |                                               |                    |                                     |                  |                                   |                    |                                     |                  |                                        |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|-----------------------------------------------|--------------------|-------------------------------------|------------------|-----------------------------------|--------------------|-------------------------------------|------------------|----------------------------------------|
| <code>hwnd</code>        | Identifies the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                       |                                                                                                                                                     |                       |                                                                                                                                                      |                          |                                               |                    |                                     |                  |                                   |                    |                                     |                  |                                        |
| <code>offset</code>      | Specifies the zero-based byte offset of the value to be retrieved. Valid values are in the range zero through the number of bytes of class memory minus two (for example, if 10 or more bytes of extra class memory was specified, a value of 8 would be an index to the fifth 16-bit integer) or one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                       |                                                                                                                                                     |                       |                                                                                                                                                      |                          |                                               |                    |                                     |                  |                                   |                    |                                     |                  |                                        |
|                          | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>GCW_CBCLSEXTRA</u></td><td>Retrieves the number of bytes of additional class information. For information about how to access this memory, see the following Comments section.</td></tr><tr><td><u>GCW_CBWNDEXTRA</u></td><td>Retrieves the number of bytes of additional window information. For information about how to access this memory, see the following Comments section.</td></tr><tr><td><u>GCW_HBRBACKGROUND</u></td><td>Retrieves the handle of the background brush.</td></tr><tr><td><u>GCW_HCURSOR</u></td><td>Retrieves the handle of the cursor.</td></tr><tr><td><u>GCW_HICON</u></td><td>Retrieves the handle of the icon.</td></tr><tr><td><u>GCW_HMODULE</u></td><td>Retrieves the handle of the module.</td></tr><tr><td><u>GCW_STYLE</u></td><td>Retrieves the window-class style bits.</td></tr></tbody></table> | Value | Meaning | <u>GCW_CBCLSEXTRA</u> | Retrieves the number of bytes of additional class information. For information about how to access this memory, see the following Comments section. | <u>GCW_CBWNDEXTRA</u> | Retrieves the number of bytes of additional window information. For information about how to access this memory, see the following Comments section. | <u>GCW_HBRBACKGROUND</u> | Retrieves the handle of the background brush. | <u>GCW_HCURSOR</u> | Retrieves the handle of the cursor. | <u>GCW_HICON</u> | Retrieves the handle of the icon. | <u>GCW_HMODULE</u> | Retrieves the handle of the module. | <u>GCW_STYLE</u> | Retrieves the window-class style bits. |
| Value                    | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                       |                                                                                                                                                     |                       |                                                                                                                                                      |                          |                                               |                    |                                     |                  |                                   |                    |                                     |                  |                                        |
| <u>GCW_CBCLSEXTRA</u>    | Retrieves the number of bytes of additional class information. For information about how to access this memory, see the following Comments section.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                       |                                                                                                                                                     |                       |                                                                                                                                                      |                          |                                               |                    |                                     |                  |                                   |                    |                                     |                  |                                        |
| <u>GCW_CBWNDEXTRA</u>    | Retrieves the number of bytes of additional window information. For information about how to access this memory, see the following Comments section.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                       |                                                                                                                                                     |                       |                                                                                                                                                      |                          |                                               |                    |                                     |                  |                                   |                    |                                     |                  |                                        |
| <u>GCW_HBRBACKGROUND</u> | Retrieves the handle of the background brush.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |                       |                                                                                                                                                     |                       |                                                                                                                                                      |                          |                                               |                    |                                     |                  |                                   |                    |                                     |                  |                                        |
| <u>GCW_HCURSOR</u>       | Retrieves the handle of the cursor.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                       |                                                                                                                                                     |                       |                                                                                                                                                      |                          |                                               |                    |                                     |                  |                                   |                    |                                     |                  |                                        |
| <u>GCW_HICON</u>         | Retrieves the handle of the icon.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |                       |                                                                                                                                                     |                       |                                                                                                                                                      |                          |                                               |                    |                                     |                  |                                   |                    |                                     |                  |                                        |
| <u>GCW_HMODULE</u>       | Retrieves the handle of the module.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                       |                                                                                                                                                     |                       |                                                                                                                                                      |                          |                                               |                    |                                     |                  |                                   |                    |                                     |                  |                                        |
| <u>GCW_STYLE</u>         | Retrieves the window-class style bits.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                       |                                                                                                                                                     |                       |                                                                                                                                                      |                          |                                               |                    |                                     |                  |                                   |                    |                                     |                  |                                        |

### Returns

The return value is the 16-bit value in the window's reserved memory, if the function is successful. Otherwise, it is zero, indicating the `hwnd` or `offset` parameter is invalid.

### Comments

To access any extra two-byte values allocated when the window-class structure was created, use a positive byte offset as the index specified by the `offset` parameter, starting at 0 for the first two-byte value in the extra space, 2 for the next two-byte value, and so on.

### See Also

GetClassInfo, GetClassLong, GetClassName, RegisterClass, SetClassWord, WNDCLASS

GCW\_CBCLSEXTRA (-20)

Retrieves the number of bytes of additional class information. For information about how to access this memory, see the following Comments section.

GCW\_CBCLSEXTRA (-20)

GCW\_CBWNDEXTRA (-18)

Retrieves the number of bytes of additional window information. For information about how to access this memory, see the following Comments section.

GCW\_CBWNDEXTRA (-18)

GCW\_HBRBACKGROUND (-10)

Retrieves the handle of the background brush.

GCW\_HBRBACKGROUND (-10)



GCW\_HCURSOR (-12)

Retrieves the handle of the cursor.

GCW\_HCURSOR (-12)

GCW\_HICON (-14)

Retrieves the handle of the icon.

GCW\_HICON (-14)

GCW\_HMODULE (-16)

Retrieves the handle of the module.

GCW\_HMODULE (-16)

GCW\_STYLE (-26)

Retrieves the window-class style bits.

GCW\_STYLE (-26)



## GetClientRect (2.x)

```
void GetClientRect(hwnd, lprc)
HWND hwnd;          /* handle of window */
RECT FAR* lprc;     /* address of
structure for rectangle */
```

The GetClientRect function retrieves the client coordinates of a window's client area. The client coordinates specify the upper-left and lower-right corners of the client area. Because client coordinates are relative to the upper-left corner of a window's client area, the coordinates of the upper-left corner are (0, 0).

| Parameter | Description                                                                                                                                                                                  |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window whose client coordinates are to be retrieved.                                                                                                                          |
| lprc      | Points to a <u>RECT</u> structure that receives the client coordinates. The left and top members will be zero. The right and bottom members will contain the width and height of the window. |

### Returns

This function does not return a value.

### See Also

GetWindowRect, RECT

## GetClipboardData (2.x)

```
HANDLE GetClipboardData(uFormat)
UINT uFormat;          /* data format */
```

The `GetClipboardData` function retrieves a handle of the current clipboard data having a specified format. The clipboard must have been opened previously.

| Parameter            | Description                                                                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>uFormat</code> | Specifies the format of the data accessed by this function. For a description of the possible data formats, see the description of the <a href="#">SetClipboardData</a> function. |

### Returns

The return value is a handle of the clipboard data in the specified format, if the function is successful. Otherwise, it is `NULL`.

### Comments

The available formats can be enumerated in advance by using the [EnumClipboardFormats](#) function.

The data handle returned by the `GetClipboardData` function is controlled by the clipboard, not by the application. The application should copy the data immediately, instead of relying on the data handle for long-term use. The application should not free the data handle or leave it locked.

Windows supports two formats for text: `CF_TEXT` (the default Windows text clipboard format) and `CF_OEMTEXT` (the format Windows uses for text in non-Windows applications). If you call `GetClipboardData` to retrieve data in one text format and the other text format is the only available text format, Windows automatically converts the text to the requested format before supplying it to your application.

If the clipboard contains data in the `CF_PALETTE` (logical color palette) format, the application should assume that any other data in the clipboard is realized against that logical palette.

### See Also

[CloseClipboard](#), [EnumClipboardFormats](#), [IsClipboardFormatAvailable](#), [OpenClipboard](#), [SetClipboardData](#)

## GetClipboardFormatName (2.x)

```
int GetClipboardFormatName(uFormat, lpszFormatName, cbMax)
UINT uFormat;          /* format to retrieve */
LPSTR lpszFormatName; /* address of buffer for name */
int cbMax;             /* length of name string */
```

The GetClipboardFormatName function retrieves the name of a registered clipboard format.

| Parameter      | Description                                                                                                             |
|----------------|-------------------------------------------------------------------------------------------------------------------------|
| uFormat        | Specifies the registered format to retrieve. This parameter must not specify any of the predefined clipboard formats.   |
| lpszFormatName | Points to a buffer that receives the format name.                                                                       |
| cbMax          | Specifies the maximum length, in bytes, of the format-name string. The format-name string is truncated if it is longer. |

### Returns

The return value is the length, in bytes, of the returned format name if the function is successful. Otherwise, it is zero, indicating the requested format does not exist or is predefined.

### See Also

[CountClipboardFormats](#), [EnumClipboardFormats](#), [GetPriorityClipboardFormat](#), [IsClipboardFormatAvailable](#), [RegisterClipboardFormat](#)

## GetClipboardOwner (2.x)

HWND GetClipboardOwner(void)

The GetClipboardOwner function retrieves the handle of the window that currently owns the clipboard, if any.

### Returns

The return value identifies the window that owns the clipboard if the function is successful. Otherwise, it is NULL.

### Comments

The clipboard can still contain data even if the clipboard is not currently owned.

### See Also

[CloseClipboard](#), [GetClipboardData](#), [GetClipboardViewer](#), [OpenClipboard](#)

## GetClipboardViewer (2.x)

HWND GetClipboardViewer(void)

The GetClipboardViewer function retrieves the handle of the first window in the clipboard-viewer chain.

### Returns

The return value identifies the window currently responsible for displaying the clipboard, if the function is successful. Otherwise, it is NULL (if there is no viewer, for example).

### See Also

[CloseClipboard](#), [GetClipboardData](#), [GetClipboardOwner](#), [OpenClipboard](#)

### GetClipCursor (3.1)

```
void GetClipCursor(lprc)  
RECT FAR* lprc;          /* address of structure for rectangle */
```

The GetClipCursor function retrieves the screen coordinates of the rectangle to which the cursor has been confined by a previous call to the ClipCursor function.

| Parameter | Description                                                                                                                                                                                          |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lprc      | Points to a <u>RECT</u> structure that receives the screen coordinates of the confining rectangle. The structure receives the dimensions of the screen if the cursor is not confined to a rectangle. |

#### Returns

This function does not return a value.

#### See Also

ClipCursor, GetCursorPos, RECT

## GetCommError (2.x)

```
int GetCommError(idComDev, lpStat)
int idComDev; /* communications device identifier */
COMSTAT FAR* lpStat;
* address of device-status buffer
/
```

The `GetCommError` function retrieves the most recent error value and current status for the specified device.

When a communications error occurs, Windows locks the communications port until `GetCommError` clears the error.

| Parameter             | Description                                                                                                                                                             |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>idComDev</code> | Specifies the communications device to be examined. The <code>OpenComm</code> function returns this value.                                                              |
| <code>lpStat</code>   | Points to the <code>COMSTAT</code> structure that is to receive the device status. If this parameter is <code>NULL</code> , the function returns only the error values. |

### Returns

The return value specifies the error value for the most recent communications-function call to the specified device, if `GetCommError` is successful.

### Errors

The return value can be a combination of the following values:

| Value                    | Meaning                                                                                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>CE_BREAK</code>    | Hardware detected a break condition.                                                                                                                                                       |
| <code>CE_CTSTO</code>    | CTS (clear-to-send) timeout. While a character was being transmitted, CTS was low for the duration specified by the <code>fCtsHold</code> member of the <code>COMSTAT</code> structure.    |
| <code>CE_DNS</code>      | Parallel device was not selected.                                                                                                                                                          |
| <code>CE_DSRTO</code>    | DSR (data-set-ready) timeout. While a character was being transmitted, DSR was low for the duration specified by the <code>fDsrHold</code> member of <code>COMSTAT</code> .                |
| <code>CE_FRAME</code>    | Hardware detected a framing error.                                                                                                                                                         |
| <code>CE_IOE</code>      | I/O error occurred during an attempt to communicate with a parallel device.                                                                                                                |
| <code>CE_MODE</code>     | Requested mode is not supported, or the <code>idComDev</code> parameter is invalid. If set, <code>CE_MODE</code> is the only valid error.                                                  |
| <code>CE_OOP</code>      | Parallel device signaled that it is out of paper.                                                                                                                                          |
| <code>CE_OVERRUN</code>  | Character was not read from the hardware before the next character arrived. The character was lost.                                                                                        |
| <code>CE_PTO</code>      | Timeout occurred during an attempt to communicate with a parallel device.                                                                                                                  |
| <code>CE_RLSDTO</code>   | RLSD (receive-line-signal-detect) timeout. While a character was being transmitted, RLSD was low for the duration specified by the <code>fRlsdHold</code> member of <code>COMSTAT</code> . |
| <code>CE_RXOVER</code>   | Receiving queue overflowed. There was either no room in the input queue or a character was received after the end-of-file character was received.                                          |
| <code>CE_RXPARITY</code> | Hardware detected a parity error.                                                                                                                                                          |
| <code>CE_TXFULL</code>   | Transmission queue was full when a function attempted to queue a character.                                                                                                                |

See Also

[OpenComm](#), [COMSTAT](#)

CE\_BREAK 0x0010

Hardware detected a break condition.



CE\_BREAK 0x0010

CE\_CTSTO 0x0020

CTS (clear-to-send) timeout. While a character was being transmitted, CTS was low for the duration specified by the fCtsHold member of the COMSTAT structure.

CE\_CTSTO 0x0020

CE\_DNS 0x0800

Parallel device was not selected.

CE\_DNS 0x0800

CE\_DSRTO 0x0040

DSR (data-set-ready) timeout. While a character was being transmitted, DSR was low for the duration specified by the fDsrHold member of COMSTAT.

CE\_DSRT0 0x0040

CE\_FRAME 0x0008

Hardware detected a framing error.



CE\_FRAME 0x0008

CE\_IOE 0x0400

I/O error occurred during an attempt to communicate with a parallel device.

CE\_IOE 0x0400

CE\_MODE 0x8000

Requested mode is not supported, or the idComDev parameter is invalid. If set, CE\_MODE is the only valid error.

CE\_MODE 0x8000

CE\_OOP 0x1000

Parallel device signaled that it is out of paper.

CE\_OOP 0x1000

CE\_OVERRUN 0x0002

Character was not read from the hardware before the next character arrived. The character was lost.



CE\_OVERRUN 0x0002

CE\_PTO 0x0200

Timeout occurred during an attempt to communicate with a parallel device.

CE\_PTO 0x0200

CE\_RLSDTO 0x0080

RLSD (receive-line-signal-detect) timeout. While a character was being transmitted, RLSD was low for the duration specified by the fRlshHold member of COMSTAT.

CE\_RLSDTO 0x0080

CE\_RXOVER 0x0001

Receiving queue overflowed. There was either no room in the input queue or a character was received after the end-of-file character was received.

CE\_RXOVER 0x0001

CE\_RXPARITY 0x0004

Hardware detected a parity error.



CE\_RXPARITY 0x0004

CE\_TXFULL 0x0100

Transmission queue was full when a function attempted to queue a character.

CE\_TXFULL 0x0100

## GetCommEventMask (2.x)

```
UINT GetCommEventMask(idComDev, fnEvtClear)
int idComDev;      /* communications device identifier */
int fnEvtClear;   /* events to clear in the event word */
/
```

The GetCommEventMask function retrieves and then clears the event word for a communications device.

| Parameter  | Description                                                                                                                                              |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| idComDev   | Specifies the communication device to be examined. The <u>OpenComm</u> function returns this value.                                                      |
| fnEvtClear | Specifies which events are to be cleared in the event word. For a list of the event values, see the description of the <u>SetCommEventMask</u> function. |

### Returns

The return value specifies the current event-word value for the specified communications device if the function is successful. Each bit in the event word specifies whether a given event has occurred; a bit is set (to 1) if the event has occurred.

### Comments

Before the GetCommEventMask function can record the occurrence of an event, an application must enable the event by using the SetCommEventMask function.

If the communication device event is a line-status or printer error, the application should call the GetCommError function after calling GetCommEventMask.

### See Also

GetCommError, OpenComm, SetCommEventMask

## GetCommState (2.x)

```
int GetCommState(idComDev, lpdcB)
int idComDev;          /* communications device identifier          */
DCB FAR* lpdcB;       /* address of structure for device control block          */
/
```

The GetCommState function retrieves the device control block for the specified device.

| Parameter | Description                                                                                                                                            |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| idComDev  | Specifies the device to be examined. The <u>OpenComm</u> function returns this value.                                                                  |
| lpdcB     | Points to the <u>DCB</u> structure that is to receive the current device control block. The DCB structure defines the control settings for the device. |

### Returns

The return value is zero if the function is successful. Otherwise, it is less than zero.

### See Also

OpenComm, SetCommState, DCB

## GetCurrentTime (2.x)

DWORD GetCurrentTime(void)

The GetCurrentTime function retrieves the number of milliseconds that have elapsed since Windows was started.

### Returns

The return value is the number of milliseconds that have elapsed since Windows was started, if the function was successful.

### Comments

The GetCurrentTime function is identical to the GetTickCount function. Applications should use the GetTickCount function, since its name matches more closely with what the function does.

### See Also

GetTickCount

## GetCursor (3.1)

HCURSOR GetCursor(void)

The GetCursor function retrieves the handle of the current cursor.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

|                                  |  |
|----------------------------------|--|
| This function has no parameters. |  |
|----------------------------------|--|

Returns

The return value is the handle of the current cursor if a cursor exists. Otherwise, it is NULL.

See Also

[SetCursor](#)

## GetCursorPos (2.x)

```
void GetCursorPos(lppt)
POINT FAR* lppt;          /* address of structure for cursor position */
```

The GetCursorPos function retrieves the screen coordinates of the cursor's current position.

| Parameter | Description                                                                                    |
|-----------|------------------------------------------------------------------------------------------------|
| lppt      | Points to the <u>POINT</u> structure that receives the cursor position, in screen coordinates. |

### Returns

This function does not return a value.

### Comments

The cursor position is always given in screen coordinates and is not affected by the mapping mode of the window that contains the cursor.

### See Also

ClipCursor, SetCursorPos, POINT



## GetDC (2.x)

```
HDC GetDC(hwnd)
HWND hwnd;    /* handle of window */
```

The GetDC function retrieves the handle of a device context for the client area of the given window. The device context can be used in subsequent graphics device interface (GDI) functions to draw in the client area.

The GetDC function retrieves a common, class, or private device context, depending on the class style specified for the given window. For common device contexts, GetDC assigns default attributes to the context each time it is retrieved. For class and private contexts, GetDC leaves the previously assigned attributes unchanged.

| Parameter | Description                                                                                                                      |
|-----------|----------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window where drawing will occur. If this parameter is NULL, the function returns a device context for the screen. |

### Returns

The return value is a handle of the device context for the given window's client area, if the function is successful. Otherwise, it is NULL.

### Comments

Unless the device context belongs to a window class, the ReleaseDC function must be called to release the context after drawing. Since only five common device contexts are available at any given time, failure to release a device context can prevent other applications from accessing a device context. If the hwnd parameter of the GetDC function is NULL, the first parameter of ReleaseDC should also be NULL.

A device context with special characteristics is returned by the GetDC function if CS\_CLASSDC, CS\_OWNDC, or CS\_PARENTDC style was specified in the WNDCLASS structure when the class was registered. For more information about these characteristics, see the description of the WNDCLASS structure.

### See Also

BeginPaint, GetDCEX, GetWindowDC, ReleaseDC, WNDCLASS

### GetDCEX (3.1)

```
HDC GetDCEX(hwnd, hrgnClip, fdwOptions)
register HWND hwnd; /* window where drawing will occur */
HRGN hrgnClip; /* clipping region that may be combined
/
DWORD fdwOptions;
* device-context options
/
```

The GetDCEX function retrieves the handle of a device context for the given window. The device context can be used in subsequent graphics device interface (GDI) functions to draw in the client area.

This function, which is an extension to the GetDC function, gives an application more control over how and whether a device context for a window is clipped.

| Parameter  | Description                                                                                               |
|------------|-----------------------------------------------------------------------------------------------------------|
| hwnd       | Identifies the window where drawing will occur.                                                           |
| hrgnClip   | Identifies a clipping region that may be combined with the visible region of the client window.           |
| fdwOptions | Specifies how the device context is created. This parameter can be a combination of the following values: |

| Value                       | Meaning                                                                                                                                                                                                                                                   |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>DCX_CACHE</u>            | Returns a device context from the cache, rather than the <u>OWNDC</u> or <u>CLASSDC</u> window. Essentially overrides <u>CS_OWNDC</u> and <u>CS_CLASSDC</u> .                                                                                             |
| <u>DCX_CLIPCHILDREN</u>     | Excludes the visible regions of all child windows below the window identified by the hwnd parameter.                                                                                                                                                      |
| <u>DCX_CLIPSIBLINGS</u>     | Excludes the visible regions of all sibling windows above the window identified by the hwnd parameter.                                                                                                                                                    |
| <u>DCX_EXCLUDERGN</u>       | Excludes the clipping region identified by the hrgnClip parameter from the visible region of the returned device context.                                                                                                                                 |
| <u>DCX_INTERSECTRGN</u>     | Intersects the clipping region identified by the hrgnClip parameter with the visible region of the returned device context.                                                                                                                               |
| <u>DCX_LOCKWINDOWUPDATE</u> | Allows drawing even if there is a <u>LockWindowUpdate</u> call in effect that would otherwise exclude this window. This value is used for drawing during tracking.                                                                                        |
| <u>DCX_PARENTCLIP</u>       | Uses the visible region of the parent window, ignoring the parent window's <u>WS_CLIPCHILDREN</u> and <u>WS_PARENTDC</u> style bits. This value sets the device context's origin to the upper-left corner of the window identified by the hwnd parameter. |
| <u>DCX_WINDOW</u>           | Returns a device context corresponding to the window rectangle rather than the client rectangle.                                                                                                                                                          |

#### Returns

The return value is a handle of the device context for the specified window, if the function is successful. Otherwise, it is NULL.

#### Comments

Unless the device context belongs to a window class, the ReleaseDC function must be called to release the context after drawing. Since only five common device contexts are available at any given time, failure to release a device context can prevent other applications from accessing a device context.

In order to obtain a cached device context, an application must specify DCX\_CACHE. If DCX\_CACHE is not specified and the window is neither CS\_OWNDC nor CS\_CLASSDC, this function returns NULL.

A device context with special characteristics is returned by the GetDC function if CS\_CLASSDC, CS\_OWNDC, or CS\_PARENTDC style was specified in the WNDCLASS structure when the class was registered. For more information about these characteristics, see the description of the WNDCLASS structure.

See Also

BeginPaint, GetDC, GetWindowDC, ReleaseDC, WNDCLASS

DCX\_CACHE 0x00000002L

Returns a device context from the cache, rather than the Owndc or ClassDc window. Essentially overrides CS\_Owndc and CS\_ClassDc.

DCX\_CACHE 0x00000002L

DCX\_CLIPCHILDREN 0x00000008L

Excludes the visible regions of all child windows below the window identified by the hwnd parameter.

DCX\_CLIPCHILDREN 0x0000008L

DCX\_CLIPSIBLINGS 0x00000010L

Excludes the visible regions of all sibling windows above the window identified by the hwnd parameter.



DCX\_CLIPSIBLINGS 0x00000010L

DCX\_EXCLUDERGN 0x00000040L

Excludes the clipping region identified by the hrgnClip parameter from the visible region of the returned device context.

DCX\_EXCLUDERGN 0x00000040L

DCX\_INTERSECTRGN 0x00000080L

Intersects the clipping region identified by the hrgnClip parameter with the visible region of the returned device context.

DCX\_INTERSECTRGN 0x0000080L

DCX\_LOCKWINDOWUPDATE 0x0000400L

Allows drawing even if there is a LockWindowUpdate call in effect that would otherwise exclude this window. This value is used for drawing during tracking.

DCX\_LOCKWINDOWUPDATE 0x0000400L

DCX\_PARENTCLIP 0x00000020L

Uses the visible region of the parent window, ignoring the parent window's WS\_CLIPCHILDREN and WS\_PARENTDC style bits. This value sets the device context's origin to the upper-left corner of the window identified by the hwnd parameter.



DCX\_PARENTCLIP 0x00000020L

DCX\_WINDOW 0x00000001L

Returns a device context corresponding to the window rectangle rather than the client rectangle.

DCX\_WINDOW 0x0000001L

## GetDesktopWindow (3.0)

HWND GetDesktopWindow(void)

The GetDesktopWindow function retrieves the handle of the desktop window. The desktop window covers the entire screen and is the area on top of which all icons and other windows are painted.

### Returns

The return value is a handle of the desktop window.

### See Also

[GetTopWindow](#), [GetWindow](#)

## GetDialogBaseUnits (3.0)

DWORD GetDialogBaseUnits(void)

The GetDialogBaseUnits function returns the dialog box base units used by Windows when creating dialog boxes. An application should use these values to calculate the average width of characters in the system font.

### Returns

The low-order word of the return value contains the width, in pixels, of the current dialog box base-width unit, if the function is successful (this base unit is derived from the system font); the high-order word of the return value contains the height, in pixels.

### Comments

The values returned represent dialog box base units before being scaled to dialog box units. The dialog box unit in the x-direction is one-fourth of the width returned by the GetDialogBaseUnits function. The dialog box unit in the y-direction is one-eighth of the height returned by the function.

To use GetDialogBaseUnits to determine the height and width, in pixels, of a control, given the width (x) and height (y) in dialog box units and the return value (IDlgBaseUnits), use the following formulas:

```
(x * LOWORD(lDlgBaseUnits)) / 4  
(y * HIWORD(lDlgBaseUnits)) / 8
```

To avoid rounding problems, perform the multiplication before the division, in case the dialog box base units are not evenly divisible by four.

### Example

The following example calculates tab stops based on the dialog box base units:

```
HMENU hmenu;  
WORD DlgWidthUnits;  
WORD TabStopList[4];  
  
case WM_CREATE:  
    hmenu = LoadMenu(hinst, "TabStopsMenu");  
    SetMenu(hwnd, hmenu);  
    DlgWidthUnits = LOWORD(GetDialogBaseUnits()) / 4;  
    TabStopList[0] = (DlgWidthUnits * 16 * 2);  
    TabStopList[1] = (DlgWidthUnits * 32 * 2);  
    TabStopList[2] = (DlgWidthUnits * 58 * 2);  
    TabStopList[3] = (DlgWidthUnits * 84 * 2);  
    break;
```

## GetDlgCtrlID (3.0)

```
int GetDlgCtrlID(hwnd)
HWND hwnd;    /* handle of child window */
```

The GetDlgCtrlID function returns a handle of a child window.

| Parameter | Description                  |
|-----------|------------------------------|
| hwnd      | Identifies the child window. |

### Returns

The return value is a handle of the child window if the function is successful. Otherwise, it is NULL.

### Comments

This function returns a handle of any child window, not just that of a control in a dialog box.

Since top-level windows do not have an identifier, the GetDlgCtrlID function's return value is invalid if the hwnd parameter identifies a top-level window.

### See Also

[GetDlgItem](#), [GetDlgItemInt](#), [GetDlgItemText](#)

## GetDlgItem (2.x)

```
HWND GetDlgItem(HWND hwndDlg, int idControl)
/* handle of dialog box */
/* identifier of control */
```

The GetDlgItem function retrieves the handle of a control that is in the given dialog box.

| Parameter | Description                                              |
|-----------|----------------------------------------------------------|
| hwndDlg   | Identifies the dialog box that contains the control.     |
| idControl | Specifies the identifier of the control to be retrieved. |

### Returns

The return value is the handle of the given control if the function is successful. Otherwise, it is NULL, indicating either an invalid dialog box handle or a nonexistent control.

### Comments

The GetDlgItem function can be used with any parent-child window pair, not just dialog boxes. As long as the hwndDlg parameter identifies a parent window and the child window has a unique identifier (as specified by the hwnd parameter in the CreateWindow function that created the child window), GetDlgItem returns the handle of the child window.

### See Also

CreateWindow, GetDlgItemID, GetDlgItemInt, GetDlgItemText, GetWindow

## GetDlgItemInt (2.x)

```
UINT GetDlgItemInt(hwndDlg, idControl, lpfTranslated, fSigned)
HWND hwndDlg;          /* handle of dialog box */
int idControl;        /* identifier of control */
BOOL FAR* lpfTranslated; /* address of variable for error flag */
BOOL fSigned;        /* signed or unsigned indicator */
```

The GetDlgItemInt function translates the text of a control in the given dialog box into an integer value.

| Parameter     | Description                                                            |
|---------------|------------------------------------------------------------------------|
| hwndDlg       | Identifies the dialog box.                                             |
| idControl     | Specifies the identifier of the dialog box control to be translated.   |
| lpfTranslated | Points to the Boolean variable that is to receive the translated flag. |
| fSigned       | Specifies whether the value to be retrieved is signed.                 |

### Returns

The return value specifies the translated value of the dialog box item text if the function is successful. Since zero is a valid return value, the lpfTranslated parameter must be used to detect errors. If an application requires a signed return value, it should cast the return value as an int type.

### Comments

The function retrieves the text of the given control by sending the control a WM\_GETTEXT message. The function then translates the text by stripping any extra spaces at the beginning of the text and converting decimal digits. The function stops translating when it reaches the end of the text or encounters a nonnumeric character. If the fSigned parameter is TRUE, the GetDlgItemInt function checks for a minus sign (–) at the beginning of the text and translates the text into a signed number. Otherwise, it creates an unsigned value.

GetDlgItemInt returns zero if the translated number is greater than 32,767 (for signed numbers) or 65,535 (for unsigned numbers). When a error occurs, such as encountering nonnumeric characters and exceeding the given maximum, GetDlgItemInt copies zero to the location pointed to by the lpfTranslated parameter. If there are no errors, lpfTranslated receives a nonzero value. If lpfTranslated is NULL, GetDlgItemInt does not warn about errors.

### See Also

GetDlgItemID, GetDlgItem, GetDlgItemText



## GetDlgItemText (2.x)

```
int GetDlgItemText(hwndDlg, idControl, lpsz, cbMax)
HWND hwndDlg;    /* handle of dialog box */
int idControl;   /* identifier of control */
LPSTR lpsz;
/* address of buffer for text
/
int cbMax;
/* maximum size of string
/
```

The GetDlgItemText function retrieves the title or text associated with a control in a dialog box.

| Parameter | Description                                                                                                                                                 |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwndDlg   | Identifies the dialog box that contains the control.                                                                                                        |
| idControl | Specifies the identifier of the control whose title is to be retrieved.                                                                                     |
| lpsz      | Points to a buffer that is to receive the control's title or text.                                                                                          |
| cbMax     | Specifies the maximum length, in bytes, of the string to be copied to the buffer pointed to by the lpsz parameter. The string is truncated if it is longer. |

### Returns

The return value specifies the number of bytes copied to the buffer, not including the terminating null character, if the function is successful. Otherwise, it is zero.

### Comments

The GetDlgItemText function sends a WM\_GETTEXT message to the control.

### See Also

GetDlgItemID, GetDlgItem, GetDlgItemInt, WM\_GETTEXT

## GetDoubleClickTime (2.x)

UINT GetDoubleClickTime(void)

The `GetDoubleClickTime` function retrieves the current double-click time for the mouse. A double-click is a series of two clicks of the mouse button, the second occurring within a specified time after the first. The double-click time is the maximum number of milliseconds that may occur between the first and second click of a double-click.

### Returns

The return value specifies the current double-click time, in milliseconds.

### See Also

[GetCapture](#), [SetDoubleClickTime](#)

### GetDriverModuleHandle (3.1)

```
HINSTANCE GetDriverModuleHandle(hdrvr)
HDRVR hdrvr; /* handle of installable driver */
```

The GetDriverModuleHandle function retrieves the instance handle of a module that contains an installable driver.

| Parameter | Description                                                                                            |
|-----------|--------------------------------------------------------------------------------------------------------|
| hdrvr     | Identifies the installable driver. This parameter must be retrieved by the <u>OpenDriver</u> function. |

#### Returns

The return value is an instance handle of the driver module if the function is successful. Otherwise, it is NULL.

#### See Also

OpenDriver

### GetDriverInfo (3.1)

```
BOOL GetDriverInfo(hdrvr, lpdis)
HDRVR hdrvr; /* handle of installable driver */
DRIVERINFOSTRUCT FAR* lpdis;
/* address of structure for info
/
```

The GetDriverInfo function retrieves information about an installable driver.

| Parameter | Description                                                                                         |
|-----------|-----------------------------------------------------------------------------------------------------|
| hdrvr     | Identifies the installable driver. This handle must be retrieved by the <u>OpenDriver</u> function. |
| lpdis     | Points to a <u>DRIVERINFOSTRUCT</u> structure that receives the driver information.                 |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

## GetFocus (2.x)

HWND GetFocus(void)

The GetFocus function retrieves the handle of the window that currently has the input focus.

### Returns

The return value is the handle of the focus window. If no window has the focus, it is NULL.

### See Also

[GetActiveWindow](#), [GetCapture](#), [SetFocus](#)

## GetFreeSystemResources (3.1)

```
UINT GetFreeSystemResources(fuSysResource)
UINT fuSysResource;          /* type of resource to check */
```

The GetFreeSystemResources function returns the percentage of free space for system resources.

| Parameter                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                             |                                                            |                          |                                                                                                                                                   |                           |                                                                                                                         |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|-----------------------------|------------------------------------------------------------|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|-------------------------------------------------------------------------------------------------------------------------|
| fuSysResource               | Specifies the type of resource to be checked. This parameter can be one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                             |                                                            |                          |                                                                                                                                                   |                           |                                                                                                                         |
|                             | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>GESR_SYSTEMRESOURCES</u></td><td>Returns the percentage of free space for system resources.</td></tr><tr><td><u>GESR_GDIRESOURCES</u></td><td>Returns the percentage of free space for GDI resources. GDI resources include device-context handles, brushes, pens, regions, fonts, and bitmaps.</td></tr><tr><td><u>GESR_USERRESOURCES</u></td><td>Returns the percentage of free space for <u>USER</u> resources. These resources include <u>window</u> and menu handles.</td></tr></tbody></table> | Value | Meaning | <u>GESR_SYSTEMRESOURCES</u> | Returns the percentage of free space for system resources. | <u>GESR_GDIRESOURCES</u> | Returns the percentage of free space for GDI resources. GDI resources include device-context handles, brushes, pens, regions, fonts, and bitmaps. | <u>GESR_USERRESOURCES</u> | Returns the percentage of free space for <u>USER</u> resources. These resources include <u>window</u> and menu handles. |
| Value                       | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                             |                                                            |                          |                                                                                                                                                   |                           |                                                                                                                         |
| <u>GESR_SYSTEMRESOURCES</u> | Returns the percentage of free space for system resources.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |                             |                                                            |                          |                                                                                                                                                   |                           |                                                                                                                         |
| <u>GESR_GDIRESOURCES</u>    | Returns the percentage of free space for GDI resources. GDI resources include device-context handles, brushes, pens, regions, fonts, and bitmaps.                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                             |                                                            |                          |                                                                                                                                                   |                           |                                                                                                                         |
| <u>GESR_USERRESOURCES</u>   | Returns the percentage of free space for <u>USER</u> resources. These resources include <u>window</u> and menu handles.                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                             |                                                            |                          |                                                                                                                                                   |                           |                                                                                                                         |

### Returns

The return value specifies the percentage of free space for resources, if the function is successful.

### Comments

Since the return value from this function does not guarantee that an application will be able to create a new object, applications should not use this function to determine whether it will be possible to create an object.

### See Also

GetFreeSpace

GFSR\_SYSTEMRESOURCES 0x0000

Returns the percentage of free space for system resources.

GFSR\_SYSTEMRESOURCES 0x0000



GFSR\_GDIRESOURCES 0x0001

Returns the percentage of free space for GDI resources. GDI resources include device-context handles, brushes, pens, regions, fonts, and bitmaps.

GFSR\_GDIRESOURCES 0x0001

GFSR\_USERRESOURCES 0x0002

Returns the percentage of free space for USER resources. These resources include window and menu handles.

GFSR\_USERRESOURCES 0x0002

## GetInputState (2.x)

BOOL GetInputState(void)

The GetInputState function determines whether there are mouse clicks or keyboard events in the system queue that require processing. Keyboard events occur when a user presses one or more keys. The system queue is the location in which Windows stores mouse and keyboard events.

### Returns

The return value is nonzero if the function detects a mouse click or keyboard event in the system queue. Otherwise, it is zero.

### See Also

[EnableHardwareInput](#)

## GetKeyboardState (2.x)

```
void GetKeyboardState(lpKeyState)
BYTE FAR* lpKeyState;          /* address of array to receive virtual-key codes */
```

The GetKeyboardState function copies the status of the 256 virtual-keyboard keys to the specified buffer.

| Parameter  | Description                                                            |
|------------|------------------------------------------------------------------------|
| lpKeyState | Points to the 256-byte buffer that will receive the virtual-key codes. |

### Returns

This function does not return a value.

### Comments

An application calls the GetKeyboardState function in response to a keyboard-input message. This function retrieves the state of the keyboard at the time the input message was generated.

If the high-order bit is 1, the key is down; otherwise, it is up. If the low-order bit is 1, the key is toggled. A toggle key, such as the CAPSLOCK key, is toggled if it has been pressed an odd number of times since the system was started. The key is untoggled if the low-order bit is 0.

### Example

The following example simulates a pressed CTRL key:

```
BYTE pbKeyState[256];

GetKeyboardState((LPBYTE) &pbKeyState);
pbKeyState[VK_CONTROL] |= 0x80;
SetKeyboardState((LPBYTE) &pbKeyState);
```

### See Also

GetKeyState, SetKeyboardState

## GetKeyState (2.x)

```
int GetKeyState(vkey)
int vkey;    /* virtual key */
```

The `GetKeyState` function retrieves the state of the specified virtual key. The state specifies whether the key is up, down, or toggled (on, off--alternating each time the key is pressed).

| Parameter         | Description                                                                                                                                                                                                                               |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>vkey</code> | Specifies a virtual key. If the requested virtual key is a letter or digit (A through Z, a through z, or 0 through 9), <code>vkey</code> must be set to the ASCII value of that character. For other keys, it must be a virtual-key code. |

### Returns

The return value specifies the state of the given virtual key. If the high-order bit is 1, the key is down; otherwise, it is up. If the low-order bit is 1, the key is toggled. A toggle key, such as the CAPSLOCK key, is toggled if it has been pressed an odd number of times since the system was started. The key is untoggled if the low-order bit is 0.

### Comments

An application calls the `GetKeyState` function in response to a keyboard-input message. This function retrieves the state of the key at the time the input message was generated.

### See Also

[GetAsyncKeyState](#), [GetKeyboardState](#)

## GetLastActivePopup (3.0)

```
HWND GetLastActivePopup(HWND hwnOwner)  
HWND hwnOwner; /* handle of owner window */
```

The GetLastActivePopup function determines which pop-up window owned by the given window was most recently active.

| Parameter | Description                  |
|-----------|------------------------------|
| hwnOwner  | Identifies the owner window. |

### Returns

The return value is the handle of most-recently active pop-up window if the function is successful.

### Comments

The return value handle will be the same as the handle in the hwnOwner parameter if any of the following conditions are met:

- The window identified by hwnOwner was most recently active.
- The window identified by hwnOwner does not own any pop-up windows.
- The window identified by hwnOwner is not a top-level window or is owned by another window.

### See Also

[AnyPopup](#), [GetActiveWindow](#), [ShowOwnedPopups](#)



## GetMenu (2.x)

```
HMENU GetMenu(hwnd)
HWND hwnd; /* handle of window */
```

The GetMenu function retrieves the handle of the menu associated with the given window.

| Parameter | Description                                           |
|-----------|-------------------------------------------------------|
| hwnd      | Identifies the window whose menu handle is retrieved. |

### Returns

The return value is the handle of the menu if the function is successful. It is NULL if the given window has no menu. It is undefined if the window is a child window.

### See Also

[GetSubMenu](#), [SetMenu](#)

## GetMenuCheckMarkDimensions (3.0)

DWORD GetMenuCheckMarkDimensions(void)

The GetMenuCheckMarkDimensions function returns the dimensions of the default check mark bitmap. Windows displays this bitmap next to checked menu items. Before calling the [SetMenuItemBitmaps](#) function to replace the default check mark, an application should determine the correct size for the bitmaps by calling the GetMenuCheckMarkDimensions function.

### Returns

The low-order word of the return value contains the width, in pixels, of the default check mark bitmap, if the function is successful; the high-order word contains the height.

### See Also

[SetMenuItemBitmaps](#)

## GetMenuItemCount (2.x)

```
int GetMenuItemCount(hmenu)
HMENU hmenu;    /* handle of menu */
```

The GetMenuItemCount function determines the number of items in a pop-up or top-level menu.

| Parameter | Description                                       |
|-----------|---------------------------------------------------|
| hmenu     | Identifies the handle of the menu to be examined. |

### Returns

The return value specifies the number of items in the menu if the function is successful. Otherwise, it is -1.

### Example

The following example initializes the items in a pop-up menu:

```
WORD wCount;
WORD wItem;
WORD wID;

case WM_INITMENUPOPUP:
    wCount = GetMenuItemCount((HMENU) wParam);
    for (wItem = 0; wItem < wCount; wItem++) {
        wID = GetMenuItemID((HMENU) wParam, wItem);
        . /* Initialize menu items. */
        .
    }
    break;
```

### See Also

GetMenu, GetMenuItemID, GetSubMenu

## GetMenuItemID (2.x)

```
UINT GetMenuItemID(hmenu, pos)
HMENU hmenu;      /* handle of menu      */
int pos;          /* position of menu item */
```

The GetMenuItemID function retrieves the identifier for a menu item located at the given position.

| Parameter | Description                                                                             |
|-----------|-----------------------------------------------------------------------------------------|
| hmenu     | Identifies the pop-up menu that contains the item whose identifier is to be retrieved.  |
| pos       | Specifies the zero-based position of the menu item whose identifier is to be retrieved. |

### Returns

The return value specifies the identifier of the pop-up menu item if the function is successful. If the hmenu parameter is NULL or if the specified item is a pop-up menu (as opposed to an item within the pop-up menu), the return value is -1. If the pos parameter corresponds to a SEPARATOR menu item, the return value is zero.

### Example

The following example initializes the items in a pop-up menu:

```
WORD wCount;
WORD wItem;
WORD wID;

case WM_INITMENUPOPUP:
    wCount = GetMenuItemCount((HMENU) wParam);
    for (wItem = 0; wItem < wCount; wItem++) {
        wID = GetMenuItemID((HMENU) wParam, wItem);
        .
        . /* Initialize menu items. */
        .
    }
    break;
```

### See Also

GetMenu, GetMenuItemCount, GetSubMenu

## GetMenuState (2.x)

```
UINT GetMenuState(hmenu, idItem, fuFlags)
HMENU hmenu;      /* handle of menu */
UINT idItem;      /* menu-item identifier */
UINT fuFlags;     /* menu flags */
/
```

The GetMenuState function retrieves the status flags associated with the specified menu item. If the menu item is a pop-up menu, this function also returns the number of items in the pop-up menu.

| Parameter     | Description                                                                                                                                                                                                                                               |       |         |              |                                     |               |                                                     |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------------|-------------------------------------|---------------|-----------------------------------------------------|
| hmenu         | Identifies the menu.                                                                                                                                                                                                                                      |       |         |              |                                     |               |                                                     |
| idItem        | Specifies the menu item for which the state is retrieved, as determined by the fuFlags parameter.                                                                                                                                                         |       |         |              |                                     |               |                                                     |
| fuFlags       | Specifies the nature of the idItem parameter. It can be one of the following values:                                                                                                                                                                      |       |         |              |                                     |               |                                                     |
|               | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MF_BYCOMMAND</td><td>Specifies the menu-item identifier.</td></tr><tr><td>MF_BYPOSITION</td><td>Specifies the zero-based position of the menu item.</td></tr></tbody></table> | Value | Meaning | MF_BYCOMMAND | Specifies the menu-item identifier. | MF_BYPOSITION | Specifies the zero-based position of the menu item. |
| Value         | Meaning                                                                                                                                                                                                                                                   |       |         |              |                                     |               |                                                     |
| MF_BYCOMMAND  | Specifies the menu-item identifier.                                                                                                                                                                                                                       |       |         |              |                                     |               |                                                     |
| MF_BYPOSITION | Specifies the zero-based position of the menu item.                                                                                                                                                                                                       |       |         |              |                                     |               |                                                     |

### Returns

The return value is -1 if the specified item does not exist. If the idItem parameter identifies a pop-up menu, the high-order byte of the return value contains the number of items in the pop-up menu, and the low order byte contains the menu flags associated with the pop-up menu. Otherwise, the return value is a mask (Boolean OR) of the values from the following list (this mask describes the status of the menu item that idItem identifies):

| Value           | Meaning                                                                                                                                                                       |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MF_BITMAP       | Item is a bitmap.                                                                                                                                                             |
| MF_CHECKED      | Check mark is placed next to item (pop-up menus only).                                                                                                                        |
| MF_DISABLED     | Item is disabled.                                                                                                                                                             |
| MF_ENABLED      | Item is enabled. Note that the value of this constant is zero; an application should not test against zero for failure when using this value.                                 |
| MF_GRAYED       | Item is disabled and grayed.                                                                                                                                                  |
| MF_MENUBARBREAK | Same as MF_MENUBREAK, except for pop-up menus where the new column is separated from the old column by a vertical dividing line.                                              |
| MF_MENUBREAK    | Item is placed on a new line (static menus) or in a new column (pop-up menus) without separating columns.                                                                     |
| MF_SEPARATOR    | Horizontal dividing line is drawn (pop-up menus only). This line cannot be enabled, checked, grayed, or highlighted. The idItem and fuFlags parameters are ignored.           |
| MF_UNCHECKED    | Check mark is not placed next to item (default). Note that the value of this constant is zero; an application should not test against zero for failure when using this value. |

### Example

The following example retrieves the handle of a pop-up menu, retrieves the checked state of a menu item in the menu, and then toggles the checked state of the item:

```
HMENU hmenu;
BOOL fOwnerDraw;

/* Retrieve a handle to the Colors menu. */
hmenu = GetSubMenu(GetMenu(hwnd), ID_COLORS_POS);

/* Retrieve the current state of the item. */

fOwnerDraw = GetMenuState(hmenu, IDM_COLOROWNERDR,
    MF_BYCOMMAND) & MF_CHECKED;
```

```
/* Toggle the state of the item. */
```

```
CheckMenuItem(hmenu, IDM_COLOROWNERDR,  
MF_BYCOMMAND | (fOwnerDraw ? MF_UNCHECKED : MF_CHECKED));
```

See Also

GetMenu, GetMenuItemCount, GetSubMenu

## GetMenuString (2.x)

```
int GetMenuString(hmenu, idItem, lpsz, cbMax, fwFlags)
HMENU hmenu;      /* handle of menu          */
UINT idItem;      /* menu-item identifier */
LPSTR lpsz;
/* address of buffer for label
/
int cbMax;
/* maximum length of label
/
UINT fwFlags;
/* menu flags
/
```

The GetMenuString function copies the label of a menu item into a buffer.

| Parameter     | Description                                                                                                                                                                                                                                               |       |         |              |                                     |               |                                                     |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------------|-------------------------------------|---------------|-----------------------------------------------------|
| hmenu         | Identifies the menu.                                                                                                                                                                                                                                      |       |         |              |                                     |               |                                                     |
| idItem        | Specifies the menu item whose label is to be copied, as determined by the fwFlags parameter.                                                                                                                                                              |       |         |              |                                     |               |                                                     |
| lpsz          | Points to a buffer that will receive the null-terminated label string.                                                                                                                                                                                    |       |         |              |                                     |               |                                                     |
| cbMax         | Specifies the maximum length, in bytes, of the label string. The label string is truncated if it is longer.                                                                                                                                               |       |         |              |                                     |               |                                                     |
| fwFlags       | Specifies the nature of the idItem parameter. It can be one of the following values:                                                                                                                                                                      |       |         |              |                                     |               |                                                     |
|               | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MF_BYCOMMAND</td><td>Specifies the menu-item identifier.</td></tr><tr><td>MF_BYPOSITION</td><td>Specifies the zero-based position of the menu item.</td></tr></tbody></table> | Value | Meaning | MF_BYCOMMAND | Specifies the menu-item identifier. | MF_BYPOSITION | Specifies the zero-based position of the menu item. |
| Value         | Meaning                                                                                                                                                                                                                                                   |       |         |              |                                     |               |                                                     |
| MF_BYCOMMAND  | Specifies the menu-item identifier.                                                                                                                                                                                                                       |       |         |              |                                     |               |                                                     |
| MF_BYPOSITION | Specifies the zero-based position of the menu item.                                                                                                                                                                                                       |       |         |              |                                     |               |                                                     |

### Returns

The return value is the length, in bytes, of the returned label, if the function is successful. The length does not include the terminating null character.

### Comments

The cbMax parameter should be one larger than the number of characters in the label to accommodate the null character that terminates the string.

### See Also

[GetMenu](#), [GetMenuItemID](#)

## GetMessage (2.x)

```
BOOL GetMessage(lpmsg, hwnd, uMsgFilterMin, uMsgFilterMax)
MSG FAR* lpmsg;          /* address of structure with message */
HWND hwnd;              /* handle of the
window
/
UINT uMsgFilterMin;
* first message
/
UINT uMsgFilterMax;
* last message
/
```

The `GetMessage` function retrieves a message from the application's message queue and places the message in a `MSG` structure. If no message is available, `GetMessage` yields control to other applications until a message becomes available.

`GetMessage` retrieves messages associated only with the given window and within the given range of message values. The function does not retrieve messages for windows that belong to other applications.

| Parameter                  | Description                                                                                                                                                                                                   |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>lpmsg</code>         | Points to an <code>MSG</code> structure that contains message information from the application's message queue.                                                                                               |
| <code>hwnd</code>          | Identifies the window whose messages are to be retrieved. If this parameter is <code>NULL</code> , <code>GetMessage</code> retrieves messages for any window that belongs to the application making the call. |
| <code>uMsgFilterMin</code> | Specifies the integer value of the lowest message value to be retrieved.                                                                                                                                      |
| <code>uMsgFilterMax</code> | Specifies the integer value of the highest message value to be retrieved.                                                                                                                                     |

### Returns

The return value is nonzero if a message other than `WM_QUIT` is retrieved. It is zero if the `WM_QUIT` message is retrieved.

### Comments

The return value is usually used to decide whether to terminate the application's main loop and exit the program.

The `WM_KEYFIRST` and `WM_KEYLAST` constants can be used as filter values to retrieve all messages related to keyboard input; the `WM_MOUSEFIRST` and `WM_MOUSELAST` constants can be used to retrieve all mouse-related messages. If the `uMsgFilterMin` and `uMsgFilterMax` parameters are both zero, the `GetMessage` function returns all available messages (without performing any filtering).

In addition to yielding control to other applications when no messages are available, the `GetMessage` and `PeekMessage` functions also yield control when `WM_PAINT` or `WM_TIMER` messages for other tasks are available.

The `GetMessage`, `PeekMessage`, and `WaitMessage` functions are the only ways to let other applications run. If your application does not call any of these functions for long periods of time, other applications cannot run.

### Example

The following example uses the `GetMessage` function to retrieve messages from a message queue, translates virtual-key messages into character messages, and dispatches messages to the appropriate window procedures:

```
MSG msg;

while (GetMessage(&msg, (HWND) NULL, 0, 0)) {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
```

### See Also

`GetMessageExtraInfo`, `PeekMessage`, `PostQuitMessage`, `SetMessageQueue`, `WaitMessage`, `MSG`, `WM_PAINT`, `WM_QUIT`, `WM_TIMER`



## GetMessageExtraInfo (3.1)

LONG GetMessageExtraInfo(void)

The GetMessageExtraInfo function retrieves the extra information associated with the last message retrieved by the GetMessage or PeekMessage function. This extra information may be added to a message by the driver for a pointing device or keyboard.

### Returns

The return value specifies the extra information if the function is successful. The meaning of the extra information is device-specific.

### See Also

GetMessage, hardware\_event, PeekMessage

## GetMessagePos (2.x)

DWORD GetMessagePos(void)

The GetMessagePos function returns a long value that represents a cursor position, in screen coordinates. This position is the point occupied by the cursor when the last message retrieved by the GetMessage function occurred.

### Returns

The return value specifies the x- and y-coordinates of the cursor position if the function is successful.

### Comments

To retrieve the current position of the cursor instead of the position at the time the last message occurred, use the GetCursorPos function.

The x-coordinate is in the low-order word of the return value; the y-coordinate is in the high-order word. If the return value is assigned to a variable, you can use the MAKEPOINT macro to obtain a POINT structure from the return value. You can also use the LOWORD or HIWORD macro to extract the x- or the y-coordinate.

### See Also

GetCursorPos, GetMessage, GetMessageTime, MAKEPOINT, LOWORD, HIWORD, POINT

## GetMessageTime (2.x)

LONG GetMessageTime(void)

The GetMessageTime function returns the message time for the last message retrieved by the GetMessage function. The time is a long integer that specifies the elapsed time, in milliseconds, from the time the system was started to the time the message was created (placed in the application queue).

### Returns

The return value specifies the message time if the function is successful.

### Comments

The return value of the GetMessageTime function does not necessarily increase between subsequent messages, because the value wraps to zero if the timer count exceeds the maximum value for long integers.

To calculate time delays between messages, verify that the time of the second message is greater than the time of the first message and then subtract the time of the first message from the time of the second message.

### See Also

GetMessage, GetMessagePos

## GetNextDlgGroupItem (2.x)

HWND GetNextDlgGroupItem(HWND hDlg, HWND hCtrl, BOOL fPrevious)

```
HWND hDlg; /* handle of dialog box */
HWND hCtrl; /* handle of control */
BOOL fPrevious;
/* direction flag
/
```

The GetNextDlgGroupItem function searches for the previous (or next) control within a group of controls in a dialog box. A group of controls begins with a control with the WS\_GROUP style and ends with the last control that does not contain a WS\_GROUP style.

| Parameter | Description                                                                                                                                                                                                                                                   |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hDlg      | Identifies the dialog box to be searched.                                                                                                                                                                                                                     |
| hCtrl     | Identifies the control to be used as the starting point for the search.                                                                                                                                                                                       |
| fPrevious | Specifies how the function is to search the group of controls in the dialog box. If this parameter is TRUE, the function searches for the previous control in the group. If this parameter is FALSE, the function searches for the next control in the group. |

### Returns

The return value is the window handle of the previous (or next) control in the group, if the function is successful.

### Comments

If the hCtrl parameter identifies the last control in the group and the fPrevious parameter is FALSE, the GetNextDlgGroupItem function returns the window handle of the first control in the group. If hCtrl identifies the first control in the group and fPrevious is TRUE, GetNextDlgGroupItem returns the window handle of the last control in the group.

### Example

The following example sets the check state of a group of radio buttons. It is assumed that the group contains only radio buttons and no other type of control:

```
HWND hwndStart, hwndCurrent;

case WM_COMMAND:
    switch (HIWORD(lParam)) {
        case BN_CLICKED:

            /*
             * If a radio button was clicked, clear the current
             * selection and select the one that was clicked.
             */

            hwndStart = GetDlgItem(hDlg, wParam);
            if (LOWORD(GetWindowLong(hwndStart,
GWL_STYLE) == BS_RADIOBUTTON)) {
                hwndCurrent = hwndStart;

                do {
                    hwndCurrent = GetNextDlgGroupItem(hDlg,
                        hwndCurrent, TRUE);
                    SendMessage(hwndCurrent, BM_SETCHECK,
                        hwndCurrent == hwndStart, 0L);
                } while (hwndCurrent != hwndStart);
            }

            /* Process other notification codes. */
        }
    }
```

### See Also

GetDlgItem, GetNextDlgTabItem

## GetNextDlgTabItem (2.x)

HWND GetNextDlgTabItem(HWND hDlg, HWND hCtrl, BOOL fPrevious)

HWND hDlg; /\* handle of dialog box \*/

HWND hCtrl;

/\* handle of known

control

/

BOOL fPrevious;

\* direction flag

/

The GetNextDlgTabItem function retrieves the handle of the first control that has the WS\_TABSTOP style that precedes (or follows) the specified control.

| Parameter | Description                                                                                                                                                                                                                                    |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hDlg      | Identifies the dialog box to be searched.                                                                                                                                                                                                      |
| hCtrl     | Identifies the control to be used as the starting point for the search.                                                                                                                                                                        |
| fPrevious | Specifies how the function is to search the dialog box. If this parameter is TRUE, the function searches for the previous control in the dialog box. If this parameter is FALSE, the function searches for the next control in the dialog box. |

### Returns

The return value is the window handle of the previous (or next) control that has the WS\_TABSTOP style, if the function is successful.

### Example

The following example retrieves the handle of the previous control that has the WS\_TABSTOP style, relative to the control that has the input focus:

```
HWND hdlg;
```

```
HWND hwndControl;
```

```
hwndControl = GetNextDlgTabItem(hdlg, GetFocus(), TRUE);
```

### See Also

GetDlgItem, GetNextDlgGroupItem

### GetNextDriver (3.1)

```
HDRVR GetNextDriver(hdrvr, fdwFlag)
HDRVR hdrvr;          /* handle of installable driver */
DWORD fdwFlag;       /* search flag */
```

The GetNextDriver function enumerates instances of an installable driver.

| Parameter | Description                                                                                                                                                                                                                                                                                                                           |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hdrvr     | Identifies the installable driver for which instances should be enumerated. This parameter must be retrieved by the <u>OpenDriver</u> function. If this parameter is NULL, the enumeration begins at either the beginning or end of the list of installable drivers (depending on the setting of the flags in the fdwFlag parameter). |
| fdwFlag   | Specifies whether the function should return a handle identifying only the first instance of a driver and whether the function should return handles identifying the instances of the driver in the order in which they were loaded. This parameter can be one or more of the following flags:                                        |

| Value                        | Meaning                                                                                                                                                                                                                                |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>GND_FIRSTINSTANCEONLY</u> | Returns a handle identifying the first instance of an installable driver. When this flag is set, the function will enumerate only the first instance of an installable driver, no matter how many times the driver has been installed. |
| <u>GND_FORWARD</u>           | Enumerates subsequent instances of the driver. (Using this flag has the same effect as not using the <u>GND_REVERSE</u> flag.)                                                                                                         |
| <u>GND_REVERSE</u>           | Enumerates instances of the driver as it was loaded--each subsequent call to the function returns the handle of the next instance.                                                                                                     |

#### Returns

The return value is the instance handle of the installable driver if the function is successful.

GND\_FIRSTINSTANCEONLY 0x00000001

Returns a handle identifying the first instance of an installable driver. When this flag is set, the function will enumerate only the first instance of an installable driver, no matter how many times the driver has been installed.

GND\_FIRSTINSTANCEONLY 0x00000001



GND\_FORWARD 0x00000000

Enumerates subsequent instances of the driver. (Using this flag has the same effect as not using the GND\_REVERSE flag.)

GND\_FORWARD 0x00000000

GND\_REVERSE 0x00000002

Enumerates instances of the driver as it was loaded--each subsequent call to the function returns the handle of the next instance.

GND\_REVERSE 0x00000002

## GetNextWindow (2.x)

```
HWND GetNextWindow(HWND hwnd, UINT uFlag)
HWND hwnd;          /* handle of current window */
UINT uFlag;         /* direction flag */
```

The `GetNextWindow` function searches for the handle of the next (or previous) window in the window manager's list. The window manager's list contains entries for all top-level windows, their associated child windows, and the child windows of any child windows. If the given window is a top-level window, the function searches for the next (or previous) handle of a top-level window. If the given window is a child window, the function searches for the handle of the next (or previous) child window.

| Parameter                | Description                                                                                                                                                                                                                                                            |       |         |                          |                                      |                          |                                          |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------------------------|--------------------------------------|--------------------------|------------------------------------------|
| <code>hwnd</code>        | Identifies the current window.                                                                                                                                                                                                                                         |       |         |                          |                                      |                          |                                          |
| <code>uFlag</code>       | Specifies whether the function should return a handle to the next window or to the previous window. It can be either of the following values:                                                                                                                          |       |         |                          |                                      |                          |                                          |
|                          | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><code>GW_HWNDNEXT</code></td><td>Returns a handle of the next window.</td></tr><tr><td><code>GW_HWNDPREV</code></td><td>Returns a handle of the previous window.</td></tr></tbody></table> | Value | Meaning | <code>GW_HWNDNEXT</code> | Returns a handle of the next window. | <code>GW_HWNDPREV</code> | Returns a handle of the previous window. |
| Value                    | Meaning                                                                                                                                                                                                                                                                |       |         |                          |                                      |                          |                                          |
| <code>GW_HWNDNEXT</code> | Returns a handle of the next window.                                                                                                                                                                                                                                   |       |         |                          |                                      |                          |                                          |
| <code>GW_HWNDPREV</code> | Returns a handle of the previous window.                                                                                                                                                                                                                               |       |         |                          |                                      |                          |                                          |

### Returns

The return value is the handle of the next (or previous) window in the window manager's list if the function is successful.

### See Also

[GetTopWindow](#), [GetWindow](#)

## GetOpenClipboardWindow (3.1)

HWND GetOpenClipboardWindow(void)

The GetOpenClipboardWindow function retrieves the handle of the window that currently has the clipboard open.

### Returns

The return value is the handle of the window that has the clipboard open, if the function is successful. Otherwise, it is NULL.

### See Also

[GetClipboardOwner](#), [GetClipboardViewer](#), [OpenClipboard](#)

## GetParent (2.x)

```
HWND GetParent(HWND  
HWND hwnd; /* handle of window */
```

The GetParent function retrieves the handle of the given window's parent window (if any).

| Parameter | Description                                                          |
|-----------|----------------------------------------------------------------------|
| hwnd      | Identifies the window whose parent window handle is to be retrieved. |

### Returns

The return value is the handle of the parent window if the function is successful. Otherwise, it is NULL, indicating an error or no parent window.

See Also

[SetParent](#)

## GetPriorityClipboardFormat (3.0)

```
int GetPriorityClipboardFormat(lpuPriorityList, cEntries)
UINT FAR* lpuPriorityList;      /* address of priority list */
int cEntries;                  /* count of entries in list */
/
```

The GetPriorityClipboardFormat function retrieves the first clipboard format in a list for which data exists in the clipboard.

| Parameter       | Description                                                                                                                                                                                          |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpuPriorityList | Points to an integer array that contains a list of clipboard formats in priority order. For a description of the data formats, see the description of the <a href="#">SetClipboardData</a> function. |
| cEntries        | Specifies the number of entries in the priority list. This value must not be greater than the number of entries in the list.                                                                         |

### Returns

The return value is the highest priority clipboard format in the list for which data exists. If no data exists in the clipboard, the return value is NULL. If data exists in the clipboard that does not match any format in the list, the return value is -1.

### See Also

[CountClipboardFormats](#), [EnumClipboardFormats](#), [GetClipboardFormatName](#), [IsClipboardFormatAvailable](#), [RegisterClipboardFormat](#), [SetClipboardData](#)



## GetProp (2.x)

```
HANDLE GetProp(hwnd, lpsz)
HWND hwnd;          /* handle of window */
LPCSTR lpsz;        /* atom or address of string */
/
```

The GetProp function retrieves a data handle from the property list of a window. The character string pointed to by the lpsz parameter identifies the handle to be retrieved. The string and handle must be added to the property list by a previous call to the SetProp function.

| Parameter | Description                                                                                                                                                                                                                                                                                                    |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window whose property list is to be searched.                                                                                                                                                                                                                                                   |
| lpsz      | Points to a null-terminated string or an atom that identifies a string. If an atom is given, it must be a global atom created by a previous call to the <u>GlobalAddAtom</u> function. The atom, a 16-bit value, must be placed in the low-order word of the lpsz parameter; the high-order word must be zero. |

### Returns

The return value is the associated data handle if the property list contains the given string. Otherwise, it is NULL.

### Comments

The value retrieved by the GetProp function can be any 16-bit value useful to the application.

### See Also

GlobalAddAtom, RemoveProp, SetProp

## GetQueueStatus (3.1)

```
DWORD GetQueueStatus(fuFlags)
UINT fuFlags; /* queue-status flags */
```

The `GetQueueStatus` function returns a value that indicates the type of messages in the queue.

This function is very fast and is typically used inside speed-critical loops to determine whether the `GetMessage` or `PeekMessage` function should be called to process input.

`GetQueueStatus` returns two sets of information: whether any new messages have been added to the queue since `GetQueueStatus`, `GetMessage`, or `PeekMessage` was last called, and what kinds of events are currently in the queue.

| Parameter                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                     |                                               |                       |                                                                                |                           |                                                    |                             |                                                   |                       |                                                |                             |                                                               |                             |                                                      |                       |                                                |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---------------------|-----------------------------------------------|-----------------------|--------------------------------------------------------------------------------|---------------------------|----------------------------------------------------|-----------------------------|---------------------------------------------------|-----------------------|------------------------------------------------|-----------------------------|---------------------------------------------------------------|-----------------------------|------------------------------------------------------|-----------------------|------------------------------------------------|
| <code>fuFlags</code>        | Specifies the queue-status flags to be retrieved. This parameter can be a combination of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                     |                                               |                       |                                                                                |                           |                                                    |                             |                                                   |                       |                                                |                             |                                                               |                             |                                                      |                       |                                                |
|                             | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><code>QS_KEY</code></td><td><code>WM_CHAR</code> message is in the queue.</td></tr><tr><td><code>QS_MOUSE</code></td><td><code>WM_MOUSEMOVE</code> or <code>WM_*BUTTON*</code> message is in the queue.</td></tr><tr><td><code>QS_MOUSEMOVE</code></td><td><code>WM_MOUSEMOVE</code> message is in the queue.</td></tr><tr><td><code>QS_MOUSEBUTTON</code></td><td><code>WM_*BUTTON*</code> message is in the queue.</td></tr><tr><td><code>QS_PAINT</code></td><td><code>WM_PAINT</code> message is in the queue.</td></tr><tr><td><code>QS_POSTMESSAGE</code></td><td>Posted message other than those listed above is in the queue.</td></tr><tr><td><code>QS_SENDMESSAGE</code></td><td>Message sent by another application is in the queue.</td></tr><tr><td><code>QS_TIMER</code></td><td><code>WM_TIMER</code> message is in the queue.</td></tr></tbody></table> | Value | Meaning | <code>QS_KEY</code> | <code>WM_CHAR</code> message is in the queue. | <code>QS_MOUSE</code> | <code>WM_MOUSEMOVE</code> or <code>WM_*BUTTON*</code> message is in the queue. | <code>QS_MOUSEMOVE</code> | <code>WM_MOUSEMOVE</code> message is in the queue. | <code>QS_MOUSEBUTTON</code> | <code>WM_*BUTTON*</code> message is in the queue. | <code>QS_PAINT</code> | <code>WM_PAINT</code> message is in the queue. | <code>QS_POSTMESSAGE</code> | Posted message other than those listed above is in the queue. | <code>QS_SENDMESSAGE</code> | Message sent by another application is in the queue. | <code>QS_TIMER</code> | <code>WM_TIMER</code> message is in the queue. |
| Value                       | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                     |                                               |                       |                                                                                |                           |                                                    |                             |                                                   |                       |                                                |                             |                                                               |                             |                                                      |                       |                                                |
| <code>QS_KEY</code>         | <code>WM_CHAR</code> message is in the queue.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                     |                                               |                       |                                                                                |                           |                                                    |                             |                                                   |                       |                                                |                             |                                                               |                             |                                                      |                       |                                                |
| <code>QS_MOUSE</code>       | <code>WM_MOUSEMOVE</code> or <code>WM_*BUTTON*</code> message is in the queue.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                     |                                               |                       |                                                                                |                           |                                                    |                             |                                                   |                       |                                                |                             |                                                               |                             |                                                      |                       |                                                |
| <code>QS_MOUSEMOVE</code>   | <code>WM_MOUSEMOVE</code> message is in the queue.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                     |                                               |                       |                                                                                |                           |                                                    |                             |                                                   |                       |                                                |                             |                                                               |                             |                                                      |                       |                                                |
| <code>QS_MOUSEBUTTON</code> | <code>WM_*BUTTON*</code> message is in the queue.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                     |                                               |                       |                                                                                |                           |                                                    |                             |                                                   |                       |                                                |                             |                                                               |                             |                                                      |                       |                                                |
| <code>QS_PAINT</code>       | <code>WM_PAINT</code> message is in the queue.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                     |                                               |                       |                                                                                |                           |                                                    |                             |                                                   |                       |                                                |                             |                                                               |                             |                                                      |                       |                                                |
| <code>QS_POSTMESSAGE</code> | Posted message other than those listed above is in the queue.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                     |                                               |                       |                                                                                |                           |                                                    |                             |                                                   |                       |                                                |                             |                                                               |                             |                                                      |                       |                                                |
| <code>QS_SENDMESSAGE</code> | Message sent by another application is in the queue.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                     |                                               |                       |                                                                                |                           |                                                    |                             |                                                   |                       |                                                |                             |                                                               |                             |                                                      |                       |                                                |
| <code>QS_TIMER</code>       | <code>WM_TIMER</code> message is in the queue.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                     |                                               |                       |                                                                                |                           |                                                    |                             |                                                   |                       |                                                |                             |                                                               |                             |                                                      |                       |                                                |

### Returns

The high-order word of the return value indicates the types of messages currently in the queue. The low-order word shows the types of messages added to the queue and are still in the queue since the last call to the `GetQueueStatus`, `GetMessage`, or `PeekMessage` function.

### Comments

The existence of a `QS_` flag in the return value does not guarantee that a subsequent call to the `PeekMessage` or `GetMessage` function will return a message. `GetMessage` and `PeekMessage` perform some internal filtering computation that may cause the message to be processed internally. For this reason, the return value from `GetQueueStatus` should be considered only a hint as to whether `GetMessage` or `PeekMessage` should be called.

### See Also

`GetInputState`, `GetMessage`, `PeekMessage`

QS\_KEY 0x0001

WM\_CHAR message is in the queue.

QS\_KEY 0x0001

QS\_MOUSE (QS\_MOUSEMOVE | QS\_MOUSEBUTTON)  
WM\_MOUSEMOVE or WM\_\*BUTTON\* message is in the queue.

QS\_MOUSE (QS\_MOUSEMOVE | QS\_MOUSEBUTTON)

QS\_MOUSEMOVE 0x0002

WM\_MOUSEMOVE message is in the queue.

QS\_MOUSEMOVE 0x0002



QS\_MOUSEBUTTON 0x0004  
WM\_\*BUTTON\* message is in the queue.

QS\_MOUSEBUTTON 0x0004

QS\_PAINT 0x0020

WM\_PAINT message is in the queue.

QS\_PAINT 0x0020

QS\_POSTMESSAGE 0x0008

Posted message other than those listed above is in the queue.

QS\_POSTMESSAGE 0x0008

QS\_SENDMESSAGE 0x0040

Message sent by another application is in the queue.

QS\_SENDMESSAGE 0x0040



QS\_TIMER 0x0010

WM\_TIMER message is in the queue.

QS\_TIMER 0x0010

## GetScrollPos (2.x)

```
int GetScrollPos(hwnd, fnBar)
HWND hwnd;    /* handle of window with scroll bar */
int fnBar;    /* scroll bar flags */
```

The GetScrollPos function retrieves the current position of the scroll box (thumb) of a scroll bar. The current position is a relative value that depends on the current scrolling range. For example, if the scrolling range is 0 through 100 and the scroll box is in the middle of the bar, the current position is 50.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |        |                                                                                                                                     |         |                                                             |         |                                                           |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------|-------------------------------------------------------------------------------------------------------------------------------------|---------|-------------------------------------------------------------|---------|-----------------------------------------------------------|
| hwnd      | Identifies a window that has standard scroll bars or a scroll bar control, depending on the value of the fnBar parameter.                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |        |                                                                                                                                     |         |                                                             |         |                                                           |
| fnBar     | Specifies the scroll bar to examine. It can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>SB_CTL</td><td>Retrieves the position of a scroll bar control. In this case, the hwnd parameter must be the window handle of a scroll bar control.</td></tr><tr><td>SB_HORZ</td><td>Retrieves the position of a window's horizontal scroll bar.</td></tr><tr><td>SB_VERT</td><td>Retrieves the position of a window's vertical scroll bar.</td></tr></tbody></table> | Value | Meaning | SB_CTL | Retrieves the position of a scroll bar control. In this case, the hwnd parameter must be the window handle of a scroll bar control. | SB_HORZ | Retrieves the position of a window's horizontal scroll bar. | SB_VERT | Retrieves the position of a window's vertical scroll bar. |
| Value     | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |        |                                                                                                                                     |         |                                                             |         |                                                           |
| SB_CTL    | Retrieves the position of a scroll bar control. In this case, the hwnd parameter must be the window handle of a scroll bar control.                                                                                                                                                                                                                                                                                                                                                                                          |       |         |        |                                                                                                                                     |         |                                                             |         |                                                           |
| SB_HORZ   | Retrieves the position of a window's horizontal scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |        |                                                                                                                                     |         |                                                             |         |                                                           |
| SB_VERT   | Retrieves the position of a window's vertical scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |        |                                                                                                                                     |         |                                                             |         |                                                           |

### Returns

The return value specifies the current position of the scroll box in the scroll bar, if the function is successful. Otherwise, it is zero, indicating that the hwnd parameter is invalid or that the window does not have a scroll bar.

### See Also

[GetScrollRange](#), [SetScrollPos](#), [SetScrollRange](#)

## GetScrollRange (2.x)

```
void GetScrollRange(HWND hwnd, int fnBar, int FAR* lpnMinPos, int FAR* lpnMaxPos)
/* handle of window with scroll bar */ /* scroll bar flags */
int FAR* lpnMinPos;
/* receives minimum position
/
int FAR* lpnMaxPos;
/* receives maximum position
/
```

The GetScrollRange function retrieves the current minimum and maximum scroll bar positions for the given scroll bar.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |        |                                                                                                                              |         |                                                             |         |                                                           |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------|------------------------------------------------------------------------------------------------------------------------------|---------|-------------------------------------------------------------|---------|-----------------------------------------------------------|
| hwnd      | Identifies a window that has standard scroll bars or a scroll bar control, depending on the value of the fnBar parameter.                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |        |                                                                                                                              |         |                                                             |         |                                                           |
| fnBar     | Specifies which scroll bar to retrieve. This parameter can be one of the following values:<br><table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>SB_CTL</td><td>Retrieves the position of a scroll bar control; in this case, the hwnd parameter must be the handle of a scroll bar control.</td></tr><tr><td>SB_HORZ</td><td>Retrieves the position of a window's horizontal scroll bar.</td></tr><tr><td>SB_VERT</td><td>Retrieves the position of a window's vertical scroll bar.</td></tr></tbody></table> | Value | Meaning | SB_CTL | Retrieves the position of a scroll bar control; in this case, the hwnd parameter must be the handle of a scroll bar control. | SB_HORZ | Retrieves the position of a window's horizontal scroll bar. | SB_VERT | Retrieves the position of a window's vertical scroll bar. |
| Value     | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |        |                                                                                                                              |         |                                                             |         |                                                           |
| SB_CTL    | Retrieves the position of a scroll bar control; in this case, the hwnd parameter must be the handle of a scroll bar control.                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |        |                                                                                                                              |         |                                                             |         |                                                           |
| SB_HORZ   | Retrieves the position of a window's horizontal scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |        |                                                                                                                              |         |                                                             |         |                                                           |
| SB_VERT   | Retrieves the position of a window's vertical scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |        |                                                                                                                              |         |                                                             |         |                                                           |
| lpnMinPos | Points to the integer variable that receives the minimum position.                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |        |                                                                                                                              |         |                                                             |         |                                                           |
| lpnMaxPos | Points to the integer variable that receives the maximum position.                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |        |                                                                                                                              |         |                                                             |         |                                                           |

### Returns

This function does not return a value.

### Comments

If the given window does not have standard scroll bars or is not a scroll bar control, the GetScrollRange function copies zero to the lpnMinPos and lpnMaxPos parameters.

The default range for a standard scroll bar is 0 through 100. The default range for a scroll bar control is empty (both values are zero).

### See Also

[GetScrollPos](#), [SetScrollPos](#), [SetScrollRange](#)

## GetSubMenu (2.x)

HMENU GetSubMenu(hmenu, nPos)

HMENU hmenu; /\* handle of menu with pop-up menu \*/

int nPos;

menu

/\* position of pop-up

\*

/

The GetSubMenu function retrieves the handle of a pop-up menu.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

|       |                                                                           |
|-------|---------------------------------------------------------------------------|
| hmenu | Identifies the menu with the pop-up menu whose handle is to be retrieved. |
|-------|---------------------------------------------------------------------------|

|      |                                                                                                                                                                                                |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nPos | Specifies the position in the given menu of the pop-up menu. Position values start at zero (zero-based) for the first menu item. The pop-up menu's identifier cannot be used in this function. |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### Returns

The return value is the handle of the given pop-up menu if the function is successful. Otherwise, it is NULL, indicating that no pop-up menu exists at the given position.

### See Also

CreatePopupMenu, GetMenu

## GetSysColor (2.x)

```
COLORREF GetSysColor(nDspElement)
int nDspElement; /* display element */
```

The GetSysColor function retrieves the current color of the specified display element. Display elements are the various parts of a window and the Windows display that appear on the system screen.

| Parameter                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------------------|-----------------------|---------------------|----------------------|--------------------|---------------------------------------------------------------------|------------------|----------|---------------|-------------------------------|--------------------|-------------------------------|-----------------|-------------------------------|---------------|-----------------------|-------------------|----------------------------------------------------------|----------------|-------------------------------------------------------------------------------------------------------------|-----------------|-------------------------------------------|---------------------|-------------------------------------|----------------------|-------------------------|-----------------------|------------------------|---------------------------|-------------------------------------|------------|------------------|----------------|----------------|-----------------|-----------------------|--------------|--------------------|-------------------|---------------|------------------|------------------|
| nDspElement               | Specifies the display element whose color is to be retrieved. This parameter can be one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
|                           | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>COLOR_ACTIVEBORDER</td><td>Active window border.</td></tr><tr><td>COLOR_ACTIVECAPTION</td><td>Active window title.</td></tr><tr><td>COLOR_APPWORKSPACE</td><td>Background color of multiple document interface (MDI) applications.</td></tr><tr><td>COLOR_BACKGROUND</td><td>Desktop.</td></tr><tr><td>COLOR_BTNFACE</td><td>Face shading on push buttons.</td></tr><tr><td>COLOR_BTNHIGHLIGHT</td><td>Selected button in a control.</td></tr><tr><td>COLOR_BTNSHADOW</td><td>Edge shading on push buttons.</td></tr><tr><td>COLOR_BTNTEXT</td><td>Text on push buttons.</td></tr><tr><td>COLOR_CAPTIONTEXT</td><td>Text in title bar, size button, scroll-bar arrow button.</td></tr><tr><td>COLOR_GRAYTEXT</td><td>Grayed (dimmed) text. This color is zero if the current display driver does not support a solid gray color.</td></tr><tr><td>COLOR_HIGHLIGHT</td><td>Background of selected item in a control.</td></tr><tr><td>COLOR_HIGHLIGHTTEXT</td><td>Text of selected item in a control.</td></tr><tr><td>COLOR_INACTIVEBORDER</td><td>Inactive window border.</td></tr><tr><td>COLOR_INACTIVECAPTION</td><td>Inactive window title.</td></tr><tr><td>COLOR_INACTIVECAPTIONTEXT</td><td>Color of text in an inactive title.</td></tr><tr><td>COLOR_MENU</td><td>Menu background.</td></tr><tr><td>COLOR_MENUTEXT</td><td>Text in menus.</td></tr><tr><td>COLOR_SCROLLBAR</td><td>Scroll-bar gray area.</td></tr><tr><td>COLOR_WINDOW</td><td>Window background.</td></tr><tr><td>COLOR_WINDOWFRAME</td><td>Window frame.</td></tr><tr><td>COLOR_WINDOWTEXT</td><td>Text in windows.</td></tr></tbody></table> | Value | Meaning | COLOR_ACTIVEBORDER | Active window border. | COLOR_ACTIVECAPTION | Active window title. | COLOR_APPWORKSPACE | Background color of multiple document interface (MDI) applications. | COLOR_BACKGROUND | Desktop. | COLOR_BTNFACE | Face shading on push buttons. | COLOR_BTNHIGHLIGHT | Selected button in a control. | COLOR_BTNSHADOW | Edge shading on push buttons. | COLOR_BTNTEXT | Text on push buttons. | COLOR_CAPTIONTEXT | Text in title bar, size button, scroll-bar arrow button. | COLOR_GRAYTEXT | Grayed (dimmed) text. This color is zero if the current display driver does not support a solid gray color. | COLOR_HIGHLIGHT | Background of selected item in a control. | COLOR_HIGHLIGHTTEXT | Text of selected item in a control. | COLOR_INACTIVEBORDER | Inactive window border. | COLOR_INACTIVECAPTION | Inactive window title. | COLOR_INACTIVECAPTIONTEXT | Color of text in an inactive title. | COLOR_MENU | Menu background. | COLOR_MENUTEXT | Text in menus. | COLOR_SCROLLBAR | Scroll-bar gray area. | COLOR_WINDOW | Window background. | COLOR_WINDOWFRAME | Window frame. | COLOR_WINDOWTEXT | Text in windows. |
| Value                     | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_ACTIVEBORDER        | Active window border.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_ACTIVECAPTION       | Active window title.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_APPWORKSPACE        | Background color of multiple document interface (MDI) applications.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_BACKGROUND          | Desktop.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_BTNFACE             | Face shading on push buttons.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_BTNHIGHLIGHT        | Selected button in a control.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_BTNSHADOW           | Edge shading on push buttons.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_BTNTEXT             | Text on push buttons.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_CAPTIONTEXT         | Text in title bar, size button, scroll-bar arrow button.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_GRAYTEXT            | Grayed (dimmed) text. This color is zero if the current display driver does not support a solid gray color.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_HIGHLIGHT           | Background of selected item in a control.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_HIGHLIGHTTEXT       | Text of selected item in a control.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_INACTIVEBORDER      | Inactive window border.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_INACTIVECAPTION     | Inactive window title.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_INACTIVECAPTIONTEXT | Color of text in an inactive title.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_MENU                | Menu background.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_MENUTEXT            | Text in menus.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_SCROLLBAR           | Scroll-bar gray area.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_WINDOW              | Window background.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_WINDOWFRAME         | Window frame.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |
| COLOR_WINDOWTEXT          | Text in windows.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                    |                       |                     |                      |                    |                                                                     |                  |          |               |                               |                    |                               |                 |                               |               |                       |                   |                                                          |                |                                                                                                             |                 |                                           |                     |                                     |                      |                         |                       |                        |                           |                                     |            |                  |                |                |                 |                       |              |                    |                   |               |                  |                  |

### Returns

The return value is a red, green, blue (RGB) color value for the specified display element, if the function is successful.

### Comments

An application can use the GetRValue, GetGValue, and GetBValue macros to extract the various colors from the return value.

### See Also

GetSystemMetrics, SetSysColors, GetRValue, GetGValue, GetBValue

## GetSysModalWindow (2.x)

HWND GetSysModalWindow(void)

The GetSysModalWindow function retrieves the handle of the system-modal window, if one is present.

### Returns

The return value is the handle of the system-modal window, if one is present. Otherwise, it is NULL.

### See Also

[SetSysModalWindow](#)

## GetSystemDebugState (3.1)

LONG GetSystemDebugState(void)

The GetSystemDebugState function retrieves information about the state of the system. A Windows-based debugger can use this information to determine whether to enter hard mode or soft mode upon encountering a breakpoint.

### Returns

The return value can be one or more of the following values:

| Value                  | Meaning                                                                                             |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <u>SDS_MENU</u>        | Menu is displayed.                                                                                  |
| <u>SDS_SYSMODAL</u>    | System-modal dialog box is displayed.                                                               |
| <u>SDS_NOTASKQUEUE</u> | Application queue does not exist yet and, therefore, the application cannot accept posted messages. |
| <u>SDS_DIALOG</u>      | Dialog box is displayed.                                                                            |
| <u>SDS_TASKLOCKED</u>  | Current task is locked and, therefore, no other task is permitted to run.                           |



SDS\_MENU 0x0001

Menu is displayed.

SDS\_MENU 0x0001

SDS\_SYSMODAL 0x0002

System-modal dialog box is displayed.

SDS\_SYSMODAL 0x0002

SDS\_NOTASKQUEUE 0x0004

Application queue does not exist yet and, therefore, the application cannot accept posted messages.

SDS\_NOTASKQUEUE 0x0004

SDS\_DIALOG 0x0008  
Dialog box is displayed.

SDS\_DIALOG 0x0008



SDS\_TASKLOCKED 0x0010

Current task is locked and, therefore, no other task is permitted to run.

SDS\_TASKLOCKED 0x0010

## GetSystemMenu (2.x)

HMENU GetSystemMenu(hwnd, fRevert)

HWND hwnd; /\* handle of window to own the System menu \*/

BOOL fRevert; /\* reset

flag

/

\*

The GetSystemMenu function allows the application to access the System menu for copying and modification.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window that will own a copy of the System menu.                                                                                                                                                                                                                                                                                                                                                                      |
| fRevert   | Specifies the action to be taken. If this parameter is FALSE, the GetSystemMenu function returns a handle of a copy of the System menu currently in use. This copy is initially identical to the System menu, but can be modified.<br>If the parameter is TRUE, GetSystemMenu resets the System menu back to the Windows default state. The previous System menu, if any, is destroyed. The return value is undefined in this case. |

### Returns

The return value is the handle of a copy of the System menu, if the fRevert parameter is FALSE. If fRevert is TRUE, the return value is undefined.

### Comments

Any window that does not use the GetSystemMenu function to make its own copy of the System menu receives the standard System menu.

The handle that GetSystemMenu returns can be used with the AppendMenu, InsertMenu, or ModifyMenu function to change the System menu. The System menu initially contains items identified by various identifier values such as SC\_CLOSE, SC\_MOVE, and SC\_SIZE. Menu items on the System menu send WM\_SYSCOMMAND messages. All predefined System-menu items have identifier numbers greater than 0xF000. If an application adds commands to the System menu, it should use identifier numbers less than 0xF000.

Windows automatically grays (dims) items on the standard System menu, depending on the situation. The application can carry out its own checking or graying by responding to the WM\_INITMENU message, which is sent before any menu is displayed.

### Example

The following example appends the About item to the System menu:

```
HMENU hmenu;
```

```
hmenu = GetSystemMenu(hwnd, FALSE);  
AppendMenu(hmenu, MF_SEPARATOR, 0, (LPSTR) NULL);  
AppendMenu(hmenu, MF_STRING, IDM_ABOUT, "About . . .");
```

### See Also

AppendMenu, InsertMenu, ModifyMenu, WM\_INITMENU

## GetSystemMetrics (2.x)

```
int GetSystemMetrics(nIndex)
```

```
int nIndex; /* system measurement to retrieve */
```

The `GetSystemMetrics` function retrieves the system metrics. The system metrics are the widths and heights of the various elements displayed by Windows. `GetSystemMetrics` can also return flags that indicate whether the current version of the Windows operating system is a debugging version, whether a mouse is present, or whether the meanings of the left and right mouse buttons have been exchanged.

| Parameter                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------------------------|---------------------------------------------|--------------------------|----------------------------------------------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|--------------------------|------------------|--------------------------|-------------------|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|--------------------------------------------------------------------|----------------------------|---------------------------------------------------------------------|-------------------------|------------------------------------------|-------------------------|-------------------------------------------|------------------------------|-------------------------------------------------------|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|------------------------|----------------|------------------------|-----------------|-------------------------------|--------------------------------------------------------------|-------------------------------|---------------------------------------------------------------|-------------------------------|-------------------------|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|--------------------------|-----------------------|---------------------------|----------------------------|-----------------------------------|----------------------------|------------------------------------|--------------------------|------------------|
| <code>nIndex</code>           | Specifies the system measurement to be retrieved. All measurements are given in pixels. The system measurement must be one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
|                               | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><code>SM_CXBORDER</code></td><td>Width of window frame that cannot be sized.</td></tr><tr><td><code>SM_CYBORDER</code></td><td>Height of window frame that cannot be sized.</td></tr><tr><td><code>SM_CYCAPTION</code></td><td>Height of window title. This is the title height plus the height of the window frame that cannot be sized (<code>SM_CYBORDER</code>).</td></tr><tr><td><code>SM_CXCURSOR</code></td><td>Width of cursor.</td></tr><tr><td><code>SM_CYCURSOR</code></td><td>Height of cursor.</td></tr><tr><td><code>SM_CXDOUBLECLK</code></td><td>Width of the rectangle around the location of the first click in a double-click sequence. The second click must occur within this rectangle for the system to consider the two clicks a double-click.</td></tr><tr><td><code>SM_CYDOUBLECLK</code></td><td>Height of the rectangle around the location of the first click in a double-click sequence. The second click must occur within this rectangle for the system to consider the two clicks a double-click.</td></tr><tr><td><code>SM_CXDLGFRAME</code></td><td>Width of frame when window has the <code>WS_DLGFRAME</code> style.</td></tr><tr><td><code>SM_CYDLGFRAME</code></td><td>Height of frame when window has the <code>WS_DLGFRAME</code> style.</td></tr><tr><td><code>SM_CXFRAME</code></td><td>Width of window frame that can be sized.</td></tr><tr><td><code>SM_CYFRAME</code></td><td>Height of window frame that can be sized.</td></tr><tr><td><code>SM_CXFULLSCREEN</code></td><td>Width of window client area for a full-screen window.</td></tr><tr><td><code>SM_CYFULLSCREEN</code></td><td>Height of window client area for a full-screen window (equivalent to the height of the screen minus the height of the window title).</td></tr><tr><td><code>SM_CXICON</code></td><td>Width of icon.</td></tr><tr><td><code>SM_CYICON</code></td><td>Height of icon.</td></tr><tr><td><code>SM_CXICONSPACING</code></td><td>Width of rectangles the system uses to position tiled icons.</td></tr><tr><td><code>SM_CYICONSPACING</code></td><td>Height of rectangles the system uses to position tiled icons.</td></tr><tr><td><code>SM_CYKANJIWINDOW</code></td><td>Height of Kanji window.</td></tr><tr><td><code>SM_CYMENU</code></td><td>Height of single-line menu bar. This is the menu height minus the height of the window frame that cannot be sized (<code>SM_CYBORDER</code>).</td></tr><tr><td><code>SM_CXMIN</code></td><td>Minimum width of window.</td></tr><tr><td><code>SM_CYMIN</code></td><td>Minimum height of window.</td></tr><tr><td><code>SM_CXMINTRACK</code></td><td>Minimum tracking width of window.</td></tr><tr><td><code>SM_CYMINTRACK</code></td><td>Minimum tracking height of window.</td></tr><tr><td><code>SM_CXSCREEN</code></td><td>Width of screen.</td></tr></tbody></table> | Value | Meaning | <code>SM_CXBORDER</code> | Width of window frame that cannot be sized. | <code>SM_CYBORDER</code> | Height of window frame that cannot be sized. | <code>SM_CYCAPTION</code> | Height of window title. This is the title height plus the height of the window frame that cannot be sized ( <code>SM_CYBORDER</code> ). | <code>SM_CXCURSOR</code> | Width of cursor. | <code>SM_CYCURSOR</code> | Height of cursor. | <code>SM_CXDOUBLECLK</code> | Width of the rectangle around the location of the first click in a double-click sequence. The second click must occur within this rectangle for the system to consider the two clicks a double-click. | <code>SM_CYDOUBLECLK</code> | Height of the rectangle around the location of the first click in a double-click sequence. The second click must occur within this rectangle for the system to consider the two clicks a double-click. | <code>SM_CXDLGFRAME</code> | Width of frame when window has the <code>WS_DLGFRAME</code> style. | <code>SM_CYDLGFRAME</code> | Height of frame when window has the <code>WS_DLGFRAME</code> style. | <code>SM_CXFRAME</code> | Width of window frame that can be sized. | <code>SM_CYFRAME</code> | Height of window frame that can be sized. | <code>SM_CXFULLSCREEN</code> | Width of window client area for a full-screen window. | <code>SM_CYFULLSCREEN</code> | Height of window client area for a full-screen window (equivalent to the height of the screen minus the height of the window title). | <code>SM_CXICON</code> | Width of icon. | <code>SM_CYICON</code> | Height of icon. | <code>SM_CXICONSPACING</code> | Width of rectangles the system uses to position tiled icons. | <code>SM_CYICONSPACING</code> | Height of rectangles the system uses to position tiled icons. | <code>SM_CYKANJIWINDOW</code> | Height of Kanji window. | <code>SM_CYMENU</code> | Height of single-line menu bar. This is the menu height minus the height of the window frame that cannot be sized ( <code>SM_CYBORDER</code> ). | <code>SM_CXMIN</code> | Minimum width of window. | <code>SM_CYMIN</code> | Minimum height of window. | <code>SM_CXMINTRACK</code> | Minimum tracking width of window. | <code>SM_CYMINTRACK</code> | Minimum tracking height of window. | <code>SM_CXSCREEN</code> | Width of screen. |
| Value                         | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CXBORDER</code>      | Width of window frame that cannot be sized.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CYBORDER</code>      | Height of window frame that cannot be sized.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CYCAPTION</code>     | Height of window title. This is the title height plus the height of the window frame that cannot be sized ( <code>SM_CYBORDER</code> ).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CXCURSOR</code>      | Width of cursor.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CYCURSOR</code>      | Height of cursor.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CXDOUBLECLK</code>   | Width of the rectangle around the location of the first click in a double-click sequence. The second click must occur within this rectangle for the system to consider the two clicks a double-click.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CYDOUBLECLK</code>   | Height of the rectangle around the location of the first click in a double-click sequence. The second click must occur within this rectangle for the system to consider the two clicks a double-click.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CXDLGFRAME</code>    | Width of frame when window has the <code>WS_DLGFRAME</code> style.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CYDLGFRAME</code>    | Height of frame when window has the <code>WS_DLGFRAME</code> style.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CXFRAME</code>       | Width of window frame that can be sized.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CYFRAME</code>       | Height of window frame that can be sized.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CXFULLSCREEN</code>  | Width of window client area for a full-screen window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CYFULLSCREEN</code>  | Height of window client area for a full-screen window (equivalent to the height of the screen minus the height of the window title).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CXICON</code>        | Width of icon.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CYICON</code>        | Height of icon.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CXICONSPACING</code> | Width of rectangles the system uses to position tiled icons.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CYICONSPACING</code> | Height of rectangles the system uses to position tiled icons.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CYKANJIWINDOW</code> | Height of Kanji window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CYMENU</code>        | Height of single-line menu bar. This is the menu height minus the height of the window frame that cannot be sized ( <code>SM_CYBORDER</code> ).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CXMIN</code>         | Minimum width of window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CYMIN</code>         | Minimum height of window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CXMINTRACK</code>    | Minimum tracking width of window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CYMINTRACK</code>    | Minimum tracking height of window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |
| <code>SM_CXSCREEN</code>      | Width of screen.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |                          |                                             |                          |                                              |                           |                                                                                                                                         |                          |                  |                          |                   |                             |                                                                                                                                                                                                       |                             |                                                                                                                                                                                                        |                            |                                                                    |                            |                                                                     |                         |                                          |                         |                                           |                              |                                                       |                              |                                                                                                                                      |                        |                |                        |                 |                               |                                                              |                               |                                                               |                               |                         |                        |                                                                                                                                                 |                       |                          |                       |                           |                            |                                   |                            |                                    |                          |                  |

|                             |                                                                                                                                                                                                                                                                                |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>SM_CYSCREEN</u>          | Height of screen.                                                                                                                                                                                                                                                              |
| <u>SM_CXHSCROLL</u>         | Width of arrow bitmap on a horizontal scroll bar.                                                                                                                                                                                                                              |
| <u>SM_CYHSCROLL</u>         | Height of arrow bitmap on a horizontal scroll bar.                                                                                                                                                                                                                             |
| <u>SM_CXVSCROLL</u>         | Width of arrow bitmap on a vertical scroll bar.                                                                                                                                                                                                                                |
| <u>SM_CYVSCROLL</u>         | Height of arrow bitmap on a vertical scroll bar.                                                                                                                                                                                                                               |
| <u>SM_CXSIZE</u>            | Width of bitmaps contained in the title bar.                                                                                                                                                                                                                                   |
| <u>SM_CYSIZE</u>            | Height of bitmaps contained in the title bar.                                                                                                                                                                                                                                  |
| <u>SM_CXHTHUMB</u>          | Width of scroll box (thumb) on horizontal scroll bar.                                                                                                                                                                                                                          |
| <u>SM_CYVTHUMB</u>          | Height of scroll box on vertical scroll bar.                                                                                                                                                                                                                                   |
| <u>SM_DBCSENABLED</u>       | Nonzero if current version of Windows uses double-byte characters; otherwise, this value returns zero.                                                                                                                                                                         |
| <u>SM_DEBUG</u>             | Nonzero if the Windows version is a debugging version.                                                                                                                                                                                                                         |
| <u>SM_MENUDROPALIGNMENT</u> | Alignment of pop-up menus. If this value is zero, the left side of a pop-up menu is aligned with the left side of the corresponding menu-bar item. If this value is nonzero, the left side of a pop-up menu is aligned with the right side of the corresponding menu-bar item. |
| <u>SM_MOUSEPRESENT</u>      | Nonzero if the mouse hardware is installed.                                                                                                                                                                                                                                    |
| <u>SM_PENWINDOWS</u>        | Handle of the Pen Windows dynamic-link library (DLL) if Pen Windows is installed.                                                                                                                                                                                              |
| <u>SM_SWAPBUTTON</u>        | Nonzero if the left and right mouse buttons are swapped.                                                                                                                                                                                                                       |

Returns

The return value specifies the requested system metric if the function is successful.

Comments

System metrics depend on the type of screen and may vary from screen to screen.

See Also

[GetSysColor](#), [SystemParametersInfo](#)

## Windows 3.1 changes

The following system-metric values have been added:

| Value                | Meaning                                                                                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SM_CXDOUBLECLK       | Width (in pixels) of the rectangle around the location of the first click in a double-click sequence. The second click must occur within this rectangle for the system to consider the two clicks a double click.                                                        |
| SM_CYDOUBLECLK       | Height (in pixels) of the rectangle around the location of the first click in a double-click sequence. The second click must occur within this rectangle for the system to consider the two clicks a double click.                                                       |
| SM_CXICONSPACING     | Width of the rectangles that the system uses to position tiled icons.                                                                                                                                                                                                    |
| SM_CYICONSPACING     | Height of the rectangles that the system uses to position tiled icons.                                                                                                                                                                                                   |
| SM_DBCSENABLED       | Nonzero if current version of Windows uses double-byte characters; otherwise, returns zero.                                                                                                                                                                              |
| SM_MENUDROPALIGNMENT | Alignment of popup menus. If this value is 0, the left side of a popup menu is aligned with the left side of the corresponding menu-bar item. If this value is nonzero, the left side of a popup menu is aligned with the right side of the corresponding menu-bar item. |
| SM_PENWINDOWS        | Handle of the Pen Windows dynamic-link library (DLL) if Pen Windows is installed.                                                                                                                                                                                        |

SM\_CXBORDER 5

Width of window frame that cannot be sized.

SM\_CXBORDER 5



SM\_CYBORDER 6

Height of window frame that cannot be sized.

SM\_CYBORDER 6

SM\_CYCAPTION 4

Height of window title. This is the title height plus the height of the window frame that cannot be sized (SM\_CYBORDER).

SM\_CYCAPTION 4

SM\_CXCURSOR 13

Width of cursor.



SM\_CYCURSOR 14

Height of cursor.

SM\_CYCURSOR 14



SM\_CXDOUBLECLK 36

Width of the rectangle around the location of the first click in a double-click sequence. The second click must occur within this rectangle for the system to consider the two clicks a double-click.

SM\_CXDOUBLECLK 36

SM\_CYDOUBLECLK 37

Height of the rectangle around the location of the first click in a double-click sequence. The second click must occur within this rectangle for the system to consider the two clicks a double-click.

SM\_CYDOUBLECLK 37

SM\_CXDLGFRAME 7

Width of frame when window has the WS\_DLGFRAME style.

SM\_CXDLGFRAME 7

SM\_CYDLGFRAME 8

Height of frame when window has the WS\_DLGFRAME style.

SM\_CYDLGFRAME 8



SM\_CXFRAME 32

Width of window frame that can be sized.

SM\_CXFRAME 32

SM\_CYFRAME 33

Height of window frame that can be sized.



SM\_CXFULLSCREEN 16

Width of window client area for a full-screen window.

SM\_CXFULLSCREEN 16

SM\_CYFULLSCREEN 17

Height of window client area for a full-screen window (equivalent to the height of the screen minus the height of the window title).

SM\_CYFULLSCREEN 17



SM\_CXICON 11

Width of icon.

SM\_CXICON 11

SM\_CYICON 12

Height of icon.

SM\_CYICON 12

SM\_CXICONSPACING 38

Width of rectangles the system uses to position tiled icons.



SM\_CYICONSPACING 39

Height of rectangles the system uses to position tiled icons.





SM\_CYKANJIWINDOW 18

Height of Kanji window.



SM\_CYMENU 15

Height of single-line menu bar. This is the menu height minus the height of the window frame that cannot be sized (SM\_CYBORDER).

SM\_CYMENU 15

SM\_CXMIN 28

Minimum width of window.



SM\_CYMIN 29

Minimum height of window.

SM\_CYMIN 29



SM\_CXMINTRACK 34

Minimum tracking width of window.

SM\_CXMINTRACK 34

SM\_CYMINTRACK 35

Minimum tracking height of window.

SM\_CYMINTRACK 35

SM\_CXSCREEN 0

Width of screen.

SM\_CXSCREEN 0

SM\_CYSCREEN 1  
Height of screen.

SM\_CYSCREEN 1



SM\_CXHSCROLL 21

Width of arrow bitmap on a horizontal scroll bar.

SM\_CXHSCROLL 21

SM\_CYHSCROLL 3

Height of arrow bitmap on a horizontal scroll bar.

SM\_CYHSCROLL 3

SM\_CXVSCROLL 2

Width of arrow bitmap on a vertical scroll bar.

SM\_CXVSCROLL 2

SM\_CYVSCROLL 20

Height of arrow bitmap on a vertical scroll bar.

SM\_CYVSCROLL 20



SM\_CXSIZE 30

Width of bitmaps contained in the title bar.

SM\_CXSIZE 30

SM\_CYSIZE 31

Height of bitmaps contained in the title bar.

SM\_CYSIZE 31

SM\_CXHTHUMB 10

Width of scroll box (thumb) on horizontal scroll bar.

SM\_CXHTHUMB 10

SM\_CYVTHUMB 9

Height of scroll box on vertical scroll bar.

SM\_CYVTHUMB 9



SM\_DBCSENABLED 42

Nonzero if current version of Windows uses double-byte characters; otherwise, this value returns zero.

SM\_DBCSENABLED 42

SM\_DEBUG 22

Nonzero if the Windows version is a debugging version.

SM\_DEBUG 22

SM\_MENUDROPALIGNMENT 40

Alignment of pop-up menus. If this value is zero, the left side of a pop-up menu is aligned with the left side of the corresponding menu-bar item. If this value is nonzero, the left side of a pop-up menu is aligned with the right side of the corresponding menu-bar item.

SM\_MENUDROPALIGNMENT 40

SM\_MOUSEPRESENT 19

Nonzero if the mouse hardware is installed.

SM\_MOUSEPRESENT 19



SM\_PENWINDOWS 41

Handle of the Pen Windows dynamic-link library (DLL) if Pen Windows is installed.



SM\_SWAPBUTTON 23

Nonzero if the left and right mouse buttons are swapped.

SM\_SWAPBUTTON 23

### GetTabbedTextExtent (3.0)

DWORD GetTabbedTextExtent(hdc, lpszString, cChars, cTabs, lpnTabs)

```
HDC hdc;          /* handle of device context          */
LPCSTR lpszString; /* address of string                                     */
/
int cChars;
/* number of characters in string
/
int cTabs;
/* number of tab positions
/
int FAR* lpnTabs;
/* address of array of tab positions
/
```

The GetTabbedTextExtent function computes the width and height of a character string. If the string contains one or more tab characters, the width of the string is based upon the specified tab stops. GetTabbedTextExtent uses the currently selected font to compute the dimensions of the string.

| Parameter  | Description                                                                                                                                                                          |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hdc        | Identifies the device context.                                                                                                                                                       |
| lpszString | Points to a character string.                                                                                                                                                        |
| cChars     | Specifies the number of characters in the text string.                                                                                                                               |
| cTabs      | Specifies the number of tab-stop positions in the array pointed to by the lpnTabs parameter.                                                                                         |
| lpnTabs    | Points to an array containing the tab-stop positions, in device units. The tab stops must be sorted in increasing order; the smallest x-value should be the first item in the array. |

#### Returns

The low-order word of the return value contains the string width, in logical units, if the function is successful; the high-order word contains the string height.

#### Comments

The current clipping region does not affect the width and height returned by the GetTabbedTextExtent function.

Since some devices do not place characters in regular cell arrays (that is, they kern the characters), the sum of the extents of the characters in a string may not be equal to the extent of the string.

If the cTabs parameter is zero and the lpnTabs parameter is NULL, tabs are expanded to eight times the average character width. If cTabs is 1, the tab stops are separated by the distance specified by the first value in the array to which lpnTabs points.

#### Example

The following example uses the LOWORD and HIWORD macros to retrieve the width and height of the string from the value returned by the GetTabbedTextExtent function:

```
LPSTR lpszTabbedText = "Column 1\tColumn 2\tTest of TabbedTextOut";
int aTabs[2] = { 150, 300 };
DWORD dwTabExtent;
WORD wStringWidth, wStringHeight;

dwTabExtent = GetTabbedTextExtent(hdc, /* handle of device context */
    lpszTabbedText, /* address of text */
    lstrlen(lpszTabbedText), /* number of characters */
    sizeof(aTabs) / sizeof(int), /* number of tabs in array */
    aTabs); /* array for tab positions */

wStringWidth = LOWORD(dwTabExtent); /* gets width of string */
wStringHeight = HIWORD(dwTabExtent); /* gets height of string */
```

#### See Also

GetTextExtent, TabbedTextOut, HIWORD, LOWORD

## GetTickCount (2.x)

DWORD GetTickCount(void)

The GetTickCount function retrieves the number of milliseconds that have elapsed since Windows was started.

### Returns

The return value specifies the number of milliseconds that have elapsed since Windows was started.

### Comments

The internal timer will wrap around to zero if Windows is run continuously for approximately 49 days.

The GetTickCount function is identical to the GetCurrentTime function. Applications should use GetTickCount, because its name matches more closely with what the function does.

### Example

The following example calls GetTickCount to determine the number of milliseconds that Windows has been running, converts the value into seconds, and displays the value in a message box:

```
char szBuf[255];

sprintf(szBuf, "Windows has been running for %lu seconds\n",
        GetTickCount() / 1000L);
MessageBox(hwnd, szBuf, "", MB_OK);
```

## GetTimerResolution (3.1)

DWORD GetTimerResolution(void)

The GetTimerResolution function retrieves the number of microseconds per timer tick.

Returns

The return value is the number of microseconds per timer tick.

See Also

[GetTickCount](#), [SetTimer](#)

## GetTopWindow (2.x)

```
HWND GetTopWindow(HWND  
HWND hwnd; /* handle of parent window */
```

The `GetTopWindow` function retrieves the handle of the top-level child window that belongs to the given parent window. If the parent window has no child windows, this function returns `NULL`.

| Parameter         | Description                                                                                                                               |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>hwnd</code> | Identifies the parent window. If this parameter is <code>NULL</code> , the function returns the first child window of the desktop window. |

### Returns

The return value is the handle of the top-level child window in a parent window's linked list of child windows. The return value is `NULL` if no child windows exist.

### See Also

[EnumWindows](#), [GetParent](#), [GetWindow](#), [IsChild](#)



## GetUpdateRect (2.x)

```
BOOL GetUpdateRect(hwnd, lprc, fErase)
HWND hwnd;          /* handle of window          */
RECT FAR* lprc;     /* address of structure for update rectangle */
/
BOOL fErase;
/* erase flag
/
```

The `GetUpdateRect` function retrieves the coordinates of the smallest rectangle that completely encloses the update region of the given window. If the window was created with the `CS_OWNDC` style and the mapping mode is not `MM_TEXT`, `GetUpdateRect` gives the rectangle in logical coordinates; otherwise, `GetUpdateRect` gives the rectangle in client coordinates. If there is no update region, `GetUpdateRect` makes the rectangle empty (sets all coordinates to zero).

| Parameter           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>hwnd</code>   | Identifies the window whose update region is to be retrieved.                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>lprc</code>   | Points to the <u>RECT</u> structure that receives the client coordinates of the enclosing rectangle.<br><br>An application can set this parameter to <code>NULL</code> to determine whether an update region exists for the window. If this parameter is <code>NULL</code> , the <code>GetUpdateRect</code> function returns nonzero if an update region exists, and zero if one does not. This provides a simple and efficient means of determining whether a <u>WM_PAINT</u> message resulted from an invalid area. |
| <code>fErase</code> | Specifies whether to erase the background in the update region. If this parameter is <code>TRUE</code> and the update region is not empty, the background is erased. To erase the background, the <code>GetUpdateRect</code> function sends a <u>WM_ERASEBKGD</u> message to the given window.                                                                                                                                                                                                                        |

### Returns

The return value is nonzero if the update region is not empty. Otherwise, it is zero.

### Comments

The update rectangle retrieved by the BeginPaint function is identical to that retrieved by the `GetUpdateRect` function.

BeginPaint automatically validates the update region, so any call to `GetUpdateRect` made immediately after the call to `BeginPaint` retrieves an empty update region.

### See Also

BeginPaint, GetUpdateRgn, InvalidateRect, UpdateWindow, ValidateRect, WM\_ERASEBKGD, RECT

### Windows 3.1 changes

An application can set the `lprc` parameter to `NULL` to determine whether an update region exists for the window. If this parameter is `NULL`, `GetUpdateRect` returns nonzero if an update region exists, and zero if one does not. This provides a simple and efficient means of determining whether a WM\_PAINT message resulted from an invalid area.

## GetUpdateRgn (2.x)

```
int GetUpdateRgn(hwnd, hrgn, fErase)
HWND hwnd;          /* handle of window */
HRGN hrgn;          /* handle of region */
BOOL fErase;       /* erase flag */
```

The GetUpdateRgn function retrieves the update region of a window. The coordinates of the update region are relative to the upper-left corner of the window (that is, they are client coordinates).

| Parameter | Description                                                                                                                                                            |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window whose update region is to be retrieved.                                                                                                          |
| hrgn      | Identifies the update region.                                                                                                                                          |
| fErase    | Specifies whether the window background should be erased and whether nonclient areas of child windows should be drawn. If this parameter is FALSE, no drawing is done. |

### Returns

The return value is SIMPLEREGION (region has no overlapping borders), COMPLEXREGION (region has overlapping borders), or NULLREGION (region is empty), if the function is successful. Otherwise, the return value is ERROR.

### Comments

The BeginPaint function automatically validates the update region, so any call to the GetUpdateRgn function made immediately after the call to BeginPaint retrieves an empty update region.

### See Also

BeginPaint, GetUpdateRect, InvalidateRgn, UpdateWindow, ValidateRgn

## GetWindow (2.x)

HWND GetWindow(HWND, fuRel)

HWND hwnd; /\* handle of original window \*/

UINT fuRel; /\* relationship flag \*/

The GetWindow function retrieves the handle of a window that has the specified relationship to the given window. The function searches the system's list of top-level windows, their associated child windows, the child windows of any child windows, and any siblings of the owner of a window.

| Parameter           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                 |                                             |                     |                                                                                                                    |                    |                                                                                                                  |                    |                                                                                        |                    |                                                                   |                 |                                |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|-----------------|---------------------------------------------|---------------------|--------------------------------------------------------------------------------------------------------------------|--------------------|------------------------------------------------------------------------------------------------------------------|--------------------|----------------------------------------------------------------------------------------|--------------------|-------------------------------------------------------------------|-----------------|--------------------------------|
| hwnd                | Identifies the original window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |                 |                                             |                     |                                                                                                                    |                    |                                                                                                                  |                    |                                                                                        |                    |                                                                   |                 |                                |
| fuRel               | Specifies the relationship between the original window and the returned window. This parameter can be one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                 |                                             |                     |                                                                                                                    |                    |                                                                                                                  |                    |                                                                                        |                    |                                                                   |                 |                                |
|                     | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>GW_CHILD</u></td><td>Identifies the window's first child window.</td></tr><tr><td><u>GW_HWNDFIRST</u></td><td>Returns the first sibling window for a child window; otherwise, it returns the first top-level window in the list.</td></tr><tr><td><u>GW_HWNDLAST</u></td><td>Returns the last sibling window for a child window; otherwise, it returns the last top-level window in the list.</td></tr><tr><td><u>GW_HWNDNEXT</u></td><td>Returns the sibling window that follows the given window in the window manager's list.</td></tr><tr><td><u>GW_HWNDPREV</u></td><td>Returns the previous sibling window in the window manager's list.</td></tr><tr><td><u>GW_OWNER</u></td><td>Identifies the window's owner.</td></tr></tbody></table> | Value | Meaning | <u>GW_CHILD</u> | Identifies the window's first child window. | <u>GW_HWNDFIRST</u> | Returns the first sibling window for a child window; otherwise, it returns the first top-level window in the list. | <u>GW_HWNDLAST</u> | Returns the last sibling window for a child window; otherwise, it returns the last top-level window in the list. | <u>GW_HWNDNEXT</u> | Returns the sibling window that follows the given window in the window manager's list. | <u>GW_HWNDPREV</u> | Returns the previous sibling window in the window manager's list. | <u>GW_OWNER</u> | Identifies the window's owner. |
| Value               | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                 |                                             |                     |                                                                                                                    |                    |                                                                                                                  |                    |                                                                                        |                    |                                                                   |                 |                                |
| <u>GW_CHILD</u>     | Identifies the window's first child window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                 |                                             |                     |                                                                                                                    |                    |                                                                                                                  |                    |                                                                                        |                    |                                                                   |                 |                                |
| <u>GW_HWNDFIRST</u> | Returns the first sibling window for a child window; otherwise, it returns the first top-level window in the list.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                 |                                             |                     |                                                                                                                    |                    |                                                                                                                  |                    |                                                                                        |                    |                                                                   |                 |                                |
| <u>GW_HWNDLAST</u>  | Returns the last sibling window for a child window; otherwise, it returns the last top-level window in the list.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                 |                                             |                     |                                                                                                                    |                    |                                                                                                                  |                    |                                                                                        |                    |                                                                   |                 |                                |
| <u>GW_HWNDNEXT</u>  | Returns the sibling window that follows the given window in the window manager's list.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |                 |                                             |                     |                                                                                                                    |                    |                                                                                                                  |                    |                                                                                        |                    |                                                                   |                 |                                |
| <u>GW_HWNDPREV</u>  | Returns the previous sibling window in the window manager's list.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                 |                                             |                     |                                                                                                                    |                    |                                                                                                                  |                    |                                                                                        |                    |                                                                   |                 |                                |
| <u>GW_OWNER</u>     | Identifies the window's owner.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                 |                                             |                     |                                                                                                                    |                    |                                                                                                                  |                    |                                                                                        |                    |                                                                   |                 |                                |

### Returns

The return value is the handle of the window if the function is successful. Otherwise, it is NULL, indicating either the end of the system's list or an invalid fuRel parameter.

### See Also

EnumWindows, FindWindow

GW\_CHILD 5

Identifies the window's first child window.

GW\_CHILD 5

GW\_HWNDFIRST 0

Returns the first sibling window for a child window; otherwise, it returns the first top-level window in the list.

GW\_HWNDFIRST 0



GW\_HWNDLAST 1

Returns the last sibling window for a child window; otherwise, it returns the last top-level window in the list.

GW\_HWNDLAST 1

GW\_HWNDNEXT 2

Returns the sibling window that follows the given window in the window manager's list.

GW\_HWNDNEXT 2

GW\_HWNDPREV 3

Returns the previous sibling window in the window manager's list.

GW\_HWNDPREV 3

GW\_OWNER 4

Identifies the window's owner.

GW\_OWNER 4



## GetWindowDC (2.x)

```
HDC GetWindowDC(hwnd)
HWND hwnd;    /* handle of window */
```

The `GetWindowDC` function retrieves a device context for the entire window, including title bar, menus, and scroll bars. A window device context permits painting anywhere in the window, because the origin of the context is the upper-left corner of the window instead of the client area.

`GetWindowDC` assigns default attributes to the device context each time it retrieves the context. Previous attributes are lost.

| Parameter | Description                                                    |
|-----------|----------------------------------------------------------------|
| hwnd      | Identifies the window whose device context is to be retrieved. |

### Returns

The return value is the handle of the device context for the given window, if the function is successful. Otherwise, it is `NULL`, indicating an error or an invalid `hwnd` parameter.

### Comments

The `GetWindowDC` function is intended to be used for special painting effects within a window's nonclient area. Painting in nonclient areas of any window is not recommended.

The `GetSystemMetrics` function can be used to retrieve the dimensions of various parts of the nonclient area, such as the title bar, menu, and scroll bars.

After painting is complete, the `ReleaseDC` function must be called to release the device context. Failure to release a window device context will have serious effects on painting requested by applications.

### See Also

[BeginPaint](#), [GetDC](#), [GetSystemMetrics](#), [ReleaseDC](#)

## GetWindowLong (2.x)

```
LONG GetWindowLong(hwnd, nOffset)
HWND hwnd;      /* handle of window */
int nOffset;    /* offset of value to retrieve */
```

The `GetWindowLong` function retrieves a long value at the specified offset into the extra window memory of the given window. Extra window memory is reserved by specifying a nonzero value in the `cbWndExtra` member of the `WNDCLASS` structure used with the `RegisterClass` function.

| Parameter            | Description                                                                                                                                                                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>hwnd</code>    | Identifies the window.                                                                                                                                                                                                                                                                                                           |
| <code>nOffset</code> | Specifies the zero-based byte offset of the value to be retrieved. Valid values are in the range zero through the number of bytes of extra window memory, minus four (for example, if 12 or more bytes of extra memory was specified, a value of 8 would be an index to the third long integer), or one of the following values: |

| Value                    | Meaning                              |
|--------------------------|--------------------------------------|
| <code>GWL_EXSTYLE</code> | Extended window style                |
| <code>GWL_STYLE</code>   | Window style                         |
| <code>GWL_WNDPROC</code> | Long pointer to the window procedure |

The following values are also available when the `hwnd` parameter identifies a dialog box:

| Value                      | Meaning                                                                                      |
|----------------------------|----------------------------------------------------------------------------------------------|
| <code>DWL_DLGPROC</code>   | Specifies the address of the dialog box procedure.                                           |
| <code>DWL_MSGRESULT</code> | Specifies the return value of a message processed in the dialog box procedure.               |
| <code>DWL_USER</code>      | Specifies extra information that is private to the application, such as handles or pointers. |

### Returns

The return value specifies information about the given window if the function is successful.

### Comments

To access any extra 4-byte values allocated when the window-class structure was created, use a positive byte offset as the index specified by the `nOffset` parameter, starting at 0 for the first 4-byte value in the extra space, 4 for the next 4-byte value, and so on.

### See Also

[GetWindowWord](#), [SetWindowLong](#), [SetWindowWord](#)

GWL\_EXSTYLE (-20)  
Extended window style

GWL\_EXSTYLE (-20)

GWL\_STYLE (-16)

Window style

GWL\_STYLE (-16)

GWL\_WNDPROC (-4)

Long pointer to the window procedure

GWL\_WNDPROC (-4)



DWL\_DLGPROC 4

Specifies the address of the dialog box procedure.

DWL\_DLGPROC 4

DWL\_MSGRESULT 0

Specifies the return value of a message processed in the dialog box procedure.

DWL\_MSGRESULT 0

DWL\_USER 8

Specifies extra information that is private to the application, such as handles or pointers.

DWL\_USER 8

### GetWindowPlacement (3.1)

```
BOOL GetWindowPlacement(HWND hwnd, WINDOWPLACEMENT FAR* lpwndpl);  
/* handle of window */  
/* address of structure for position data */
```

The GetWindowPlacement function retrieves the show state and the normal (restored), minimized, and maximized positions of a window.

| Parameter | Description                                                                                           |
|-----------|-------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window.                                                                                |
| lpwndpl   | Points to the <u>WINDOWPLACEMENT</u> structure that receives the show state and position information. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The flags member of the WINDOWPLACEMENT structure retrieved by this function is always zero. If the window identified by the hwnd parameter is maximized, the showCmd member of WINDOWPLACEMENT is SW\_SHOWMAXIMIZED; if the window is minimized, it is SW\_SHOWMINIMIZED; and it is SW\_SHOWNORMAL otherwise.

#### See Also

SetWindowPlacement, WINDOWPLACEMENT

## GetWindowRect (2.x)

```
void GetWindowRect(hwnd, lprc)
HWND hwnd;          /* handle of window */
RECT FAR* lprc;    /* address of structure for window coordinates */
```

The GetWindowRect function retrieves the dimensions of the bounding rectangle of a given window. The dimensions are given in screen coordinates, relative to the upper-left corner of the display screen, and include the title bar, border, and scroll bars, if present.

| Parameter | Description                                                                                                                     |
|-----------|---------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window.                                                                                                          |
| lprc      | Points to a <u>RECT</u> structure that receives the screen coordinates of the upper-left and lower-right corners of the window. |

### Returns

This function does not return a value.

### Example

The following example calls the GetWindowRect function to retrieve the dimensions of the desktop window, and uses the dimensions to create a window that fills the right third of the desktop window:

```
RECT rc;
WORD wWidth;

GetWindowRect(GetDesktopWindow(), &rc);

/* Set the width to be 1/3 of the desktop window's width. */
wWidth = (rc.right - rc.left) / 3;

/* Create a main window for this application instance. */

hwndFrame = CreateWindow("MyClass", "My Title", WS_OVERLAPPEDWINDOW,
    rc.right - wWidth, /* horizontal position */
    0, /* vertical position */
    wWidth, /* width */
    rc.bottom, /* height */
    (HWND) NULL, (HMENU) NULL, hinst, (LPSTR) NULL);
```

### See Also

GetClientRect, MoveWindow, SetWindowPos, RECT



## GetWindowTask (2.x)

```
HTASK GetWindowTask(hwnd)
HWND hwnd;    /* handle of window */
```

The GetWindowTask function searches for the handle of a task associated with a window. A task is any program that executes as an independent unit. All applications are executed as tasks. Each instance of an application is a task.

| Parameter | Description                                                |
|-----------|------------------------------------------------------------|
| hwnd      | Identifies the window for which to retrieve a task handle. |

### Returns

The return value is the handle of the task associated with a particular window, if the function is successful. Otherwise, it is NULL.

### See Also

[EnumTaskWindows](#), [GetCurrentTask](#)

## GetWindowText (2.x)

```
int GetWindowText(hwnd, lpsz, cbMax)
HWND hwnd;      /* handle of window */
LPSTR lpsz;     /* address of buffer for text */
int cbMax;
/* maximum number of bytes to copy */
```

The GetWindowText function copies text of the given window's title bar (if it has one) into a buffer. If the given window is a control, the text within the control is copied.

| Parameter | Description                                                                                                                                                            |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window or control containing the title bar or text.                                                                                                     |
| lpsz      | Points to a buffer that will receive the title bar or text.                                                                                                            |
| cbMax     | Specifies the maximum number of characters to copy to the buffer. The title bar or text is truncated if it is longer than the number of characters specified in cbMax. |

### Returns

The return value specifies the length, in bytes, of the copied string, not including the terminating null character. It is zero if the window has no title bar, the title bar is empty, or the hwnd parameter is invalid.

### Comments

This function causes a WM\_GETTEXT message to be sent to the given window or control.

### See Also

GetWindowTextLength, WM\_GETTEXT

## GetWindowTextLength (2.x)

```
int GetWindowTextLength(hwnd)
HWND hwnd;    /* handle of window with text */
```

The `GetWindowTextLength` function retrieves the length, in bytes, of the text in the given window's title bar. If the window is a control, the length of the text within the control is retrieved.

| Parameter         | Description                       |
|-------------------|-----------------------------------|
| <code>hwnd</code> | Identifies the window or control. |

### Returns

The return value specifies the text length, in bytes, not including any null terminating character, if the function is successful. Otherwise, it is zero.

### Comments

This function causes the `WM_GETTEXTLENGTH` message to be sent to the given window or control.

### See Also

`GetWindowText`, `WM_GETTEXT`, `WM_GETTEXTLENGTH`

## GetWindowWord (2.x)

```
WORD GetWindowWord(hwnd, nOffset)
HWND hwnd;      /* handle of window */
int nOffset;    /* offset of value to retrieve */
```

The `GetWindowWord` function retrieves a word value at the specified offset into the extra window memory of the given window. Extra window memory is reserved by specifying a nonzero value in the `cbWndExtra` member of the `WNDCLASS` structure used with the `RegisterClass` function.

| Parameter                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                            |                                                                   |                             |                                                                                                                                                                                                                                                    |                     |                                               |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|----------------------------|-------------------------------------------------------------------|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|-----------------------------------------------|
| <code>hwnd</code>           | Identifies the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |                            |                                                                   |                             |                                                                                                                                                                                                                                                    |                     |                                               |
| <code>nOffset</code>        | Specifies the zero-based byte offset of the value to be retrieved. Valid values are in the range zero through the number of bytes of extra window memory, minus two (for example, if 10 or more bytes of extra memory was specified, a value of 8 would be an index to the fifth integer), or one of the following values:                                                                                                                                                                                                                                                                                    |       |         |                            |                                                                   |                             |                                                                                                                                                                                                                                                    |                     |                                               |
|                             | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><code>GWW_HINSTANCE</code></td><td>Specifies the instance handle of the module that owns the window.</td></tr><tr><td><code>GWW_HWNDPARENT</code></td><td>Specifies the handle of the parent window, if any. The <code>SetParent</code> function changes the parent window of a child window. An application should not call the <code>SetWindowWord</code> function to change the parent of a child window.</td></tr><tr><td><code>GWW_ID</code></td><td>Specifies the identifier of the child window.</td></tr></tbody></table> | Value | Meaning | <code>GWW_HINSTANCE</code> | Specifies the instance handle of the module that owns the window. | <code>GWW_HWNDPARENT</code> | Specifies the handle of the parent window, if any. The <code>SetParent</code> function changes the parent window of a child window. An application should not call the <code>SetWindowWord</code> function to change the parent of a child window. | <code>GWW_ID</code> | Specifies the identifier of the child window. |
| Value                       | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                            |                                                                   |                             |                                                                                                                                                                                                                                                    |                     |                                               |
| <code>GWW_HINSTANCE</code>  | Specifies the instance handle of the module that owns the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                            |                                                                   |                             |                                                                                                                                                                                                                                                    |                     |                                               |
| <code>GWW_HWNDPARENT</code> | Specifies the handle of the parent window, if any. The <code>SetParent</code> function changes the parent window of a child window. An application should not call the <code>SetWindowWord</code> function to change the parent of a child window.                                                                                                                                                                                                                                                                                                                                                            |       |         |                            |                                                                   |                             |                                                                                                                                                                                                                                                    |                     |                                               |
| <code>GWW_ID</code>         | Specifies the identifier of the child window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |                            |                                                                   |                             |                                                                                                                                                                                                                                                    |                     |                                               |

### Returns

The return value specifies information about the given window if the function is successful.

### Comments

To access any extra two-byte values allocated when the window-class structure was created, use a positive byte offset as the index specified by the `nOffset` parameter, starting at 0 for the first two-byte value in the extra space, 2 for the next two-byte value, and so on.

### See Also

[GetWindowLong](#), [SetParent](#), [SetWindowLong](#), [SetWindowWord](#)

GWW\_HINSTANCE (-6)

Specifies the instance handle of the module that owns the window.

GWW\_HINSTANCE (-6)

GWW\_HWNDPARENT (-8)

Specifies the handle of the parent window, if any. The `SetParent` function changes the parent window of a child window. An application should not call the `SetWindowWord` function to change the parent of a child window.

GWW\_HWNDPARENT (-8)



GWW\_ID (-12)

Specifies the identifier of the child window.

GWW\_ID (-12)

## GlobalAddAtom (2.x)

```
ATOM GlobalAddAtom(lpszString)
LPCSTR lpszString;      /* address of string to add */
```

The GlobalAddAtom function adds a string to the system atom table and returns a unique value identifying the string.

| Parameter  | Description                                                                                                                                                                                                                  |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszString | Points to the null-terminated string to be added. The case of the first string added is preserved and returned by the <a href="#">GlobalGetAtomName</a> function. Strings that differ only in case are considered identical. |

### Returns

The return value identifies the string if the function is successful. Otherwise, it is zero.

### Comments

If the string exists already in the system atom table, the atom for the existing string will be returned and the atom's reference count will be incremented (increased by one). The string associated with the atom will not be deleted from memory until its reference count is zero. For more information, see the description of the [GlobalDeleteAtom](#) function.

Global atoms are not deleted automatically when the application terminates. For every call to the GlobalAddAtom function, there must be a corresponding call to the [GlobalDeleteAtom](#) function.

### Example

The following example adds the string "This is a global atom" to the system atom table:

```
ATOM atom;
char szMsg[80];

atom = GlobalAddAtom("This is a global atom");

if (atom == 0)
    MessageBox(hwnd, "GlobalAddAtom failed", "",
    MB_ICONSTOP);
else {
    wsprintf(szMsg, "GlobalAddAtom returned %u", atom);
    MessageBox(hwnd, szMsg, "", MB_OK);
}
```

### See Also

[AddAtom](#), [GlobalDeleteAtom](#), [GlobalGetAtomName](#)

## GlobalDeleteAtom (2.x)

```
ATOM GlobalDeleteAtom(atm)
ATOM atm; /* atom to delete */
```

The GlobalDeleteAtom function decrements (decreases by one) the reference count of a global atom. If the atom's reference count reaches zero, the string associated with the atom is removed from the system atom table.

| Parameter | Description                        |
|-----------|------------------------------------|
| atom      | Identifies the atom to be deleted. |

### Returns

The return value is zero if the function is successful. The return value is equal to the atom parameter if the function failed to decrement the reference count for the specified atom.

### Comments

An atom's reference count specifies the number of times the string has been added to the atom table. The [GlobalAddAtom](#) function increments (increases by one) the reference count each time it is called with a string that already exists in the system atom table.

The only way to ensure that an atom has been deleted from the atom table is to call this function repeatedly until it fails. When the count is decremented to zero, the next [GlobalFindAtom](#) or [GlobalDeleteAtom](#) function call will fail.

### Example

The following example repeatedly calls the GlobalDeleteAtom function to decrement the reference count for the atom until the atom is deleted and the GlobalDeleteAtom function does not return zero:

```
int cRef;
ATOM atom;
char szMsg[80];

for (cRef = 0; ((atom = GlobalFindAtom("This is a global atom")) != 0)
;
    cRef++)
    GlobalDeleteAtom(atom);

wsprintf(szMsg, "reference count was %d", cRef);
MessageBox(hwnd, szMsg, "GlobalDeleteAtom", MB\_OK);
```

### See Also

[DeleteAtom](#), [GlobalAddAtom](#), [GlobalFindAtom](#)

## GlobalFindAtom (2.x)

ATOM GlobalFindAtom(lpszString)  
LPCSTR lpszString; /\* address of string to find \*/

The GlobalFindAtom function searches the system atom table for the specified character string and retrieves the global atom associated with that string. (A global atom is an atom that is available to all Windows applications.)

| Parameter  | Description                                                   |
|------------|---------------------------------------------------------------|
| lpszString | Points to the null-terminated character string to search for. |

### Returns

The return value identifies the global atom associated with the given string, if the function is successful. Otherwise, if the string is not in the table, the return value is zero.

### Example

The following example repeatedly calls the GlobalFindAtom function to retrieve the atom associated with the string "This is a global atom". The example uses the [GlobalDeleteAtom](#) function to decrement (decrease by one) the reference count for the atom until the atom is deleted and GlobalFindAtom returns zero.

```
int cRef;  
ATOM atom;  
char szMsg[80];  
  
for (cRef = 0; ((atom = GlobalFindAtom("This is a global atom")) != 0)  
; cRef++)  
    GlobalDeleteAtom(atom);  
  
wsprintf(szMsg, "reference count was %d", cRef);  
MessageBox(hwnd, szMsg, "GlobalDeleteAtom", MB_OK);
```

### See Also

[FindAtom](#), [GlobalAddAtom](#), [GlobalDeleteAtom](#)

## GlobalGetAtomName (2.x)

```
UINT GlobalGetAtomName(atom, lpzBuffer, cbBuffer)
ATOM atom;          /* atom identifier */
LPSTR lpzBuffer;    /* address of
buffer for atom string
/
int cbBuffer;
* size of buffer
/
```

The GlobalGetAtomName function retrieves a copy of the character string associated with the given global atom. (A global atom is an atom that is available to all Windows applications.)

| Parameter | Description                                                                      |
|-----------|----------------------------------------------------------------------------------|
| atom      | Identifies the global atom associated with the character string to be retrieved. |
| lpzBuffer | Points to the buffer for the character string.                                   |
| cbBuffer  | Specifies the size, in bytes, of the buffer.                                     |

### Returns

The return value specifies the number of bytes copied to the buffer, not including the null-terminating character, if the function is successful.

### Example

The following example uses the GlobalGetAtomName function to retrieve the character string associated with a global atom:

```
char szBuf[80];

GlobalGetAtomName(atGlobal, szBuf, sizeof(szBuf));

MessageBox(hwnd, szBuf, "GlobalGetAtomName", MB_OK);
```

See Also

GetAtomName

## GrayString (2.x)

```
BOOL GrayString(hdc, hbr, gsprc, lParam, cch, x, y, cx, cy)
HDC hdc; /* handle of device context */
HBRUSH hbr; /* handle of brush for graying */
GRAYSTRINGPROC gsprc; /* address of callback function */
LPARAM lParam; /* address of application-defined data */
int cch; /* number of characters to output */
int x; /* horizontal position */
int y; /* vertical position */
int cx; /* width */
int cy; /* height */
```

The `GrayString` function draws gray (dim) text at the given location by writing the text in a memory bitmap, graying the bitmap, and then copying the bitmap to the display. The function grays the text regardless of the selected brush and background. `GrayString` uses the font currently selected for the given device context.

| Parameter           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>hdc</code>    | Identifies the device context.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>hbr</code>    | Identifies the brush to be used for graying.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>gsprc</code>  | Specifies the procedure-instance address of the application-supplied callback function that will draw the string. The address must be created by the <a href="#">MakeProcInstance</a> function. For more information about the callback function, see the description of the <a href="#">GrayStringProc</a> callback function.<br>If this parameter is <code>NULL</code> , the system uses the <a href="#">TextOut</a> function to draw the string, and the <code>lParam</code> parameter is assumed to be a long pointer to the character string to be output. |
| <code>lParam</code> | Points to data to be passed to the output function. If the <code>gsprc</code> parameter is <code>NULL</code> , the <code>lParam</code> parameter must point to the string to be output.                                                                                                                                                                                                                                                                                                                                                                         |
| <code>cch</code>    | Specifies the number of characters to be output. If this parameter is zero, the <code>GrayString</code> function calculates the length of the string (assuming that the <code>lParam</code> parameter is a pointer to the string). If <code>cch</code> is -1 and the function pointed to by the <code>gsprc</code> parameter returns zero, the image is shown but not grayed.                                                                                                                                                                                   |
| <code>x</code>      | Specifies the logical x-coordinate of the starting position of the rectangle that encloses the string.                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>y</code>      | Specifies the logical y-coordinate of the starting position of the rectangle that encloses the string.                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>cx</code>     | Specifies the width, in logical units, of the rectangle that encloses the string. If this parameter is zero, the <code>GrayString</code> function calculates the width of the area, assuming the <code>lParam</code> parameter is a pointer to the string.                                                                                                                                                                                                                                                                                                      |
| <code>cy</code>     | Specifies the height, in logical units, of the rectangle that encloses the string. If this parameter is zero, the <code>GrayString</code> function calculates the height of the area, assuming the <code>lParam</code> parameter is a pointer to the string.                                                                                                                                                                                                                                                                                                    |

Returns

The return value is nonzero if the function is successful. It is zero if either the TextOut function or the application-supplied output function returns zero, or if there is insufficient memory to create a memory bitmap for graying.

#### Comments

An application must select the MM\_TEXT mapping mode before using this function.

If TextOut cannot handle the string to be output (for example, if the string is stored as a bitmap), the gsprc parameter must point to a callback function that will draw the string.

An application can draw grayed strings on devices that support a solid gray color without calling the GrayString function. The system color COLOR\_GRAYTEXT is the solid-gray system color used to draw disabled text. The application can call the GetSysColor function to retrieve the color value of COLOR\_GRAYTEXT. If the color is other than zero (black), the application can call the SetTextColor function to set the text color to the color value and then draw the string directly. If the retrieved color is black, the application must call GrayString to gray the text.

#### See Also

GetSysColor, MakeProcInstance, SetTextColor, TextOut



### hardware\_event (3.1)

```
extrn hardware_event :far
```

```
mov  ax, Msg      ; message
mov  cx, ParamL   ; low-order word of lParam of the message
mov  dx, ParamH   ; high-order word of lParam of the message
mov  si, hwnd     ; handle of the destination window
mov  di, wParam   ; wParam of the message
cCall hardware_event
```

The hardware\_event function places a hardware-related message into the system message queue. This function allows a driver for a non-standard hardware device to place a message into the queue.

| Parameter | Description                                                                                                                                                                                                                                                                   |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Msg       | Specifies the message to place in the system message queue.                                                                                                                                                                                                                   |
| ParamL    | Specifies the low-order word of the lParam parameter of the message.                                                                                                                                                                                                          |
| lParamH   | Specifies the high-order word of the lParam parameter of the message.                                                                                                                                                                                                         |
| hwnd      | Identifies the window to which the message is directed. This parameter also becomes the low-order word of the dwExtraInfo parameter associated with the message. An application can determine the value of this parameter by calling the <u>GetMessageExtraInfo</u> function. |
| wParam    | Specifies the wParam parameter of the message.                                                                                                                                                                                                                                |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

An application should not use this function to place keyboard or mouse messages into the system message queue.

An application may only call the hardware\_event function from an assembly language routine. The application must declare the function as follows:

```
extrn hardware_event :far
```

If the application includes CMACROS.INC, the application can declare the function as follows:

```
extrnFP hardware_event.
```

#### See Also

GetMessageExtraInfo, MOUSEHOOKSTRUCT

## HideCaret (2.x)

```
void HideCaret(hwnd)  
HWND hwnd; /* handle of window with caret */
```

The HideCaret function hides the caret by removing it from the screen. Although the caret is no longer visible, it can be displayed again by using the [ShowCaret](#) function. Hiding the caret does not destroy its current shape.

| Parameter | Description                                                                                                                                            |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window that owns the caret. This parameter can be set to NULL to specify indirectly the window in the current task that owns the caret. |

### Returns

This function does not return a value.

### Comments

The HideCaret function hides the caret only if the given window owns the caret. If the hwnd parameter is NULL, the function hides the caret only if a window in the current task owns the caret.

Hiding is cumulative. If HideCaret has been called five times in a row, [ShowCaret](#) must be called five times before the caret will be shown.

### See Also

[CreateCaret](#), [ShowCaret](#)

## HiliteMenuItem (2.x)

```
BOOL HiliteMenuItem(hwnd, hmenu, idHiliteItem, fuHilite)
HWND hwnd; /* handle of window with menu */
HMENU hmenu; /* handle of menu */
UINT idHiliteItem;
/* menu-item identifier */
UINT fuHilite;
/* highlight flags */
```

The HiliteMenuItem function highlights or removes the highlighting from a top-level (menu-bar) menu item.

| Parameter     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |              |                                                                                               |               |                                                                                  |           |                                                                                                   |             |                                             |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------------|-----------------------------------------------------------------------------------------------|---------------|----------------------------------------------------------------------------------|-----------|---------------------------------------------------------------------------------------------------|-------------|---------------------------------------------|
| hwnd          | Identifies the window that contains the menu.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |              |                                                                                               |               |                                                                                  |           |                                                                                                   |             |                                             |
| hmenu         | Identifies the top-level menu that contains the item to be highlighted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |              |                                                                                               |               |                                                                                  |           |                                                                                                   |             |                                             |
| idHiliteItem  | Specifies the menu item to be highlighted, as determined by the fuHilite parameter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |              |                                                                                               |               |                                                                                  |           |                                                                                                   |             |                                             |
| fuHilite      | Specifies whether the menu item is highlighted or the highlight is removed. It can be a combination of the MF_HILITE or MF_UNHILITE value with the MF_BYCOMMAND or MF_BYPOSITION value. These values have the following meanings:                                                                                                                                                                                                                                                                                                                                      |       |         |              |                                                                                               |               |                                                                                  |           |                                                                                                   |             |                                             |
|               | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MF_BYCOMMAND</td><td>Menu-item identifier is specified by the idHiliteItem parameter (the default interpretation).</td></tr><tr><td>MF_BYPOSITION</td><td>Zero-based position of the menu item is specified by the idHiliteItem parameter.</td></tr><tr><td>MF_HILITE</td><td>Menu item is highlighted. If this value is not given, highlighting is removed from the menu item.</td></tr><tr><td>MF_UNHILITE</td><td>Highlighting is removed from the menu item.</td></tr></tbody></table> | Value | Meaning | MF_BYCOMMAND | Menu-item identifier is specified by the idHiliteItem parameter (the default interpretation). | MF_BYPOSITION | Zero-based position of the menu item is specified by the idHiliteItem parameter. | MF_HILITE | Menu item is highlighted. If this value is not given, highlighting is removed from the menu item. | MF_UNHILITE | Highlighting is removed from the menu item. |
| Value         | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |              |                                                                                               |               |                                                                                  |           |                                                                                                   |             |                                             |
| MF_BYCOMMAND  | Menu-item identifier is specified by the idHiliteItem parameter (the default interpretation).                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |              |                                                                                               |               |                                                                                  |           |                                                                                                   |             |                                             |
| MF_BYPOSITION | Zero-based position of the menu item is specified by the idHiliteItem parameter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |              |                                                                                               |               |                                                                                  |           |                                                                                                   |             |                                             |
| MF_HILITE     | Menu item is highlighted. If this value is not given, highlighting is removed from the menu item.                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |              |                                                                                               |               |                                                                                  |           |                                                                                                   |             |                                             |
| MF_UNHILITE   | Highlighting is removed from the menu item.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |              |                                                                                               |               |                                                                                  |           |                                                                                                   |             |                                             |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The MF\_HILITE and MF\_UNHILITE flags can be used only with the HiliteMenuItem function; they cannot be used with the ModifyMenu function.

### See Also

CheckMenuItem, EnableMenuItem, ModifyMenu

## InflateRect (2.x)

```
void InflateRect(lprc, xAmt, yAmt)
RECT FAR* lprc;          /* address of rectangle */
int xAmt;                /* amount to
increase or decrease width
/
int yAmt;
* amount to increase or decrease height
/
```

The InflateRect function increases or decreases the width and height of a rectangle. The InflateRect function adds xAmt units to the left and right ends of the rectangle and adds yAmt units to the top and bottom. The xAmt and yAmt parameters are signed values; positive values increase the width and height, and negative values decrease them.

| Parameter | Description                                                                                                    |
|-----------|----------------------------------------------------------------------------------------------------------------|
| lprc      | Points to the <u>RECT</u> structure that increases or decreases in size.                                       |
| xAmt      | Specifies the amount to increase or decrease the rectangle width. It must be negative to decrease the width.   |
| yAmt      | Specifies the amount to increase or decrease the rectangle height. It must be negative to decrease the height. |

### Returns

This function does not return a value.

### Comments

The width and height of a rectangle must not be greater than 32,767 units or less than -32,768 units.

### See Also

IntersectRect, OffsetRect, UnionRect, RECT

## InSendMessage (2.x)

BOOL InSendMessage(void)

The InSendMessage function specifies whether the current window procedure is processing a message that was sent from another task by a call to the SendMessage function.

### Returns

The return value is nonzero if the window procedure is processing a message sent to it from another task by the SendMessage function. Otherwise, the return value is zero.

### Comments

Applications use the InSendMessage function to determine how to handle errors that occur when an inactive window processes messages. For example, if the active window uses the SendMessage function to send a request for information to another window, the other window cannot become active until it returns control from the SendMessage call. The only method an inactive window has to inform the user of an error is to create a message box.

### See Also

PostAppMessage, SendMessage

## InsertMenu (3.0)

```
BOOL InsertMenu(hmenu, idItem, fuFlags, idNewItem, lpNewItem)
HMENU hmenu;          /* handle of menu */
UINT idItem;          /* menu item that new menu item is to precede */
UINT fuFlags;        /* menu flags */
UINT idNewItem;      /* item identifier or pop-up menu handle */
LPCSTR lpNewItem;    /* item content */
```

The InsertMenu function inserts a new menu item into a menu, moving other items down the menu. The function also sets the state of the menu item.

| Parameter     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |              |                                                          |               |                                                                                                                                                 |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------------|----------------------------------------------------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| hmenu         | Identifies the menu to be changed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |              |                                                          |               |                                                                                                                                                 |
| idItem        | Specifies the menu item before which the new menu item is to be inserted, as determined by the fuFlags parameter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |              |                                                          |               |                                                                                                                                                 |
| fuFlags       | Specifies how the idItem parameter is interpreted and information about the state of the new menu item when it is added to the menu. This parameter consists of a combination of one of the following values and the values listed in the Comments section.<br><table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MF_BYCOMMAND</td><td>The idItem parameter specifies the menu-item identifier.</td></tr><tr><td>MF_BYPOSITION</td><td>The idItem parameter specifies the zero-based position of the menu item. If idItem is -1, the new menu item is appended to the end of the menu.</td></tr></tbody></table>                                                                                                    | Value | Meaning | MF_BYCOMMAND | The idItem parameter specifies the menu-item identifier. | MF_BYPOSITION | The idItem parameter specifies the zero-based position of the menu item. If idItem is -1, the new menu item is appended to the end of the menu. |
| Value         | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |              |                                                          |               |                                                                                                                                                 |
| MF_BYCOMMAND  | The idItem parameter specifies the menu-item identifier.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |              |                                                          |               |                                                                                                                                                 |
| MF_BYPOSITION | The idItem parameter specifies the zero-based position of the menu item. If idItem is -1, the new menu item is appended to the end of the menu.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |              |                                                          |               |                                                                                                                                                 |
| idNewItem     | Specifies either the identifier of the new menu item or, if fuFlags is set to <u>MF_POPUP</u> , the menu handle of the pop-up menu.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |              |                                                          |               |                                                                                                                                                 |
| lpNewItem     | Specifies the contents of the new menu item. If fuFlags is set to <u>MF_STRING</u> (the default value), this parameter points to a null-terminated string. If fuFlags is set to <u>MF_BITMAP</u> instead, lpNewItem contains a bitmap handle in its low-order word. If fuFlags is set to <u>MF_OWNERDRAW</u> , lpNewItem specifies an application-defined 32-bit value, which the application can use to maintain additional data associated with the menu item. This 32-bit value is available to the application in the itemData member of the structure pointed to by the lParam parameter of the <u>WM_MEASUREITEM</u> and <u>WM_DRAWITEM</u> messages. These messages are sent when the menu item is initially displayed or is changed. |       |         |              |                                                          |               |                                                                                                                                                 |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

If the active multiple document interface (MDI) child window is maximized and an application inserts a pop-up menu into the MDI application's menu by calling this function and specifying the MF\_BYPOSITION flag, the menu is inserted one position farther left than expected. This occurs because the System menu of the active MDI child window is inserted into the first position of the MDI frame window's menu bar. To avoid this behavior, the application must add 1 to the position value that would otherwise be used. An application can use the WM\_MDIGETACTIVE message to determine whether the currently active child window is maximized.

Whenever a menu changes (whether or not the menu is in a window that is displayed), the application should call the DrawMenuBar function.

Each of the following groups lists flags that should not be used together:

- MF\_BYCOMMAND and MF\_BYPOSITION
- MF\_DISABLED, MF\_ENABLED, and MF\_GRAYED
- MF\_BITMAP, MF\_STRING, MF\_OWNERDRAW, and MF\_SEPARATOR

- MF\_MENUBARBREAK and MF\_MENUBREAK
- MF\_CHECKED and MF\_UNCHECKED

The following list describes the flags that may be set in the fuFlags parameter:

| Value           | Meaning                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MF_BITMAP       | Uses a bitmap as the item. The low-order word of the lpNewItem parameter contains the handle of the bitmap.                                                                                                                                                                                                                                                                                                          |
| MF_BYCOMMAND    | Specifies that the idItem parameter gives the menu-item identifier (default).                                                                                                                                                                                                                                                                                                                                        |
| MF_BYPOSITION   | Specifies that the idItem parameter gives the position of the menu item rather than the menu-item identifier.                                                                                                                                                                                                                                                                                                        |
| MF_CHECKED      | Places a check mark next to (selects) the menu item. If the application has supplied check-mark bitmaps (see the <u>SetMenuItemBitmaps</u> function), setting this flag displays the check-mark bitmap next to the menu item.                                                                                                                                                                                        |
| MF_DISABLED     | Disables the menu item so that it cannot be selected, but does not gray (dim) it.                                                                                                                                                                                                                                                                                                                                    |
| MF_ENABLED      | Enables the menu item so that it can be selected, and restores it from its grayed state.                                                                                                                                                                                                                                                                                                                             |
| MF_GRAYED       | Disables the menu item so that it cannot be selected, and grays it.                                                                                                                                                                                                                                                                                                                                                  |
| MF_MENUBARBREAK | Same as <u>MF_MENUBREAK</u> except, for pop-up menus, separates the new column from the old column by using a vertical line.                                                                                                                                                                                                                                                                                         |
| MF_MENUBREAK    | Places the menu item on a new line for static menu-bar items. For pop-up menus, places the menu item in a new column, with no dividing line between the columns.                                                                                                                                                                                                                                                     |
| MF_OWNERDRAW    | Specifies that the item is an owner-drawn item. The window that owns the menu receives a <u>WM_MEASUREITEM</u> message (when the menu is displayed for the first time) to retrieve the height and width of the menu item. The <u>WM_DRAWITEM</u> message is then sent to the owner whenever the owner <u>must update the visual appearance</u> of the menu item. This option is not valid for a top-level menu item. |
| MF_POPUP        | Specifies that the menu item has a pop-up menu associated with it. The idNewItem parameter specifies a handle of a pop-up menu to be associated with the item. Use the <u>MF_OWNERDRAW</u> flag to add either a top-level pop-up menu or a hierarchical pop-up menu to a pop-up menu item.                                                                                                                           |
| MF_SEPARATOR    | Draws a horizontal dividing line. You can use this flag in a pop-up menu. This line cannot be grayed, disabled, or highlighted. Windows ignores the lpNewItem and idNewItem parameters.                                                                                                                                                                                                                              |
| MF_STRING       | Specifies that the menu item is a character string; the lpNewItem parameter points to the string for the item.                                                                                                                                                                                                                                                                                                       |
| MF_UNCHECKED    | Does not place a check mark next to the item (default value). If the application has supplied check-mark bitmaps (see <u>SetMenuItemBitmaps</u> ), setting this flag displays the check-mark-off bitmap next to the menu item.                                                                                                                                                                                       |

See Also

AppendMenu, CreateMenu, DrawMenuBar, RemoveMenu, SetMenuItemBitmaps,  
DRAWITEMSTRUCT, MEASUREITEMSTRUCT, WM\_DRAWITEM, WM\_MDIGETACTIVE,  
WM\_MEASUREITEM

## IntersectRect (2.x)

```
BOOL IntersectRect(lprcDst, lprcSrc1, lprcSrc2)
RECT FAR* lprcDst;          /* address of structure for intersection */
const RECT FAR* lprcSrc1;   /* address of structure with 1st rectangle */
/
const RECT FAR* lprcSrc2;   /* address of structure with 2nd rectangle */
/
```

The IntersectRect function calculates the intersection of two source rectangles and places the coordinates of the intersection rectangle into the destination rectangle. If the rectangles do not intersect, an empty rectangle (0, 0, 0, 0) is placed into the destination rectangle.

| Parameter | Description                                                                                                                            |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------|
| lprcDst   | Points to a <u>RECT</u> structure that receives the intersection of the rectangles pointed to by the lprcSrc1 and lprcSrc2 parameters. |
| lprcSrc1  | Points to the <u>RECT</u> structure that contains the first source rectangle.                                                          |
| lprcSrc2  | Points to the <u>RECT</u> structure that contains the second source rectangle.                                                         |

### Returns

The return value is nonzero if the rectangles intersect. Otherwise, it is zero.

### See Also

InflateRect, SubtractRect, UnionRect, RECT



## InvalidateRect (2.x)

```
void InvalidateRect(hwnd, lprc, fErase)
HWND hwnd; /* handle of window with changed update region */
const RECT FAR* lprc;
/* address of structure with rectangle
/
BOOL fErase;
* erase-background flag
/
```

The `InvalidateRect` function adds a rectangle to a window's update region. The update region represents the client area of the window that must be redrawn.

| Parameter | Description                                                                                                                                                                                                                                                                                                       |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window whose update region has changed.                                                                                                                                                                                                                                                            |
| lprc      | Points to a <code>RECT</code> structure that contains the client coordinates of the rectangle to be added to the update region. If the <code>lprc</code> parameter is <code>NULL</code> , the entire client area is added to the update region.                                                                   |
| fErase    | Specifies whether the background within the update region is to be erased when the update region is processed. If this parameter is <code>TRUE</code> , the background is erased when the <code>BeginPaint</code> function is called. If this parameter is <code>FALSE</code> , the background remains unchanged. |

### Returns

This function does not return a value.

### Comments

The invalidated areas accumulate in the update region until the region is processed when the next `WM_PAINT` message occurs, or until the region is validated by using the `ValidateRect` or `ValidateRgn` function.

Windows sends a `WM_PAINT` message to a window whenever its update region is not empty and there are no other messages in the application queue for that window.

If the `fErase` parameter is `TRUE` for any part of the update region, the background is erased in the entire region, not just in the given part.

### See Also

[BeginPaint](#), [InvalidateRgn](#), [ValidateRect](#), [ValidateRgn](#), [RECT](#), [WM\\_PAINT](#)

## InvalidateRgn (2.x)

```
void InvalidateRgn(hwnd, hrgn, fErase)
HWND hwnd; /* handle of window with changed update region */
HRGN hrgn; /* handle of region to add */
BOOL fErase;
/* erase-background flag */
```

The `InvalidateRgn` function adds a region to a window's update region. The update region represents the client area of the window that must be redrawn.

| Parameter | Description                                                                                                                                                                                                                                                                                                      |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window whose update region has changed.                                                                                                                                                                                                                                                           |
| hrgn      | Identifies the region to be added to the update region. The region is assumed to have client coordinates. If this parameter is <code>NULL</code> , the entire client area is added to the update region.                                                                                                         |
| fErase    | Specifies whether the background within the update region is to be erased when the update region is processed. If this parameter is <code>TRUE</code> , the background is erased when the <code>BeginPaint</code> function is called. If the parameter is <code>FALSE</code> , the background remains unchanged. |

### Returns

This function does not return a value.

### Comments

The invalidated regions accumulate in the update region until the region is processed when the next `WM_PAINT` message occurs, or until the region is validated by using the `ValidateRect` or `ValidateRgn` function.

Windows sends a `WM_PAINT` message to a window whenever its update region is not empty and there are no other messages in the application queue for that window.

If the `fErase` parameter is `TRUE` for any part of the update region, the background is erased in the entire region, not just in the given part.

### See Also

[BeginPaint](#), [InvalidateRect](#), [ValidateRect](#), [ValidateRgn](#), [WM\\_PAINT](#)

## InvertRect (2.x)

```
void InvertRect(hdc, lprc)
HDC hdc;          /* handle of device context */
const RECT FAR* lprc; /* address of structure with rectangle */
/
```

The InvertRect function inverts a rectangular area. Inversion is a logical NOT operation and flips the bits of each pixel.

| Parameter | Description                                                                                              |
|-----------|----------------------------------------------------------------------------------------------------------|
| hdc       | Identifies the device context.                                                                           |
| lprc      | Points to a <u>RECT</u> structure that contains the logical coordinates of the rectangle to be inverted. |

### Returns

This function does not return a value.

### Comments

On monochrome screens, the InvertRect function makes white pixels black and black pixels white. On color screens, the inversion depends on how colors are generated for the screen. Calling InvertRect twice, specifying the same rectangle, restores the display to its previous colors.

The InvertRect function compares the values of the top, bottom, left, and right members of the specified rectangle. If bottom is less than or equal to top, or if right is less than or equal to left, the function does not draw the rectangle.

### See Also

FillRect, RECT

## IsCharAlpha (3.0)

```
BOOL IsCharAlpha(chTest)
char chTest;      /* character to test */
```

The IsCharAlpha function determines whether a character is in the set of language-defined alphabetic characters.

| Parameter | Description                           |
|-----------|---------------------------------------|
| chTest    | Specifies the character to be tested. |

### Returns

The return value is nonzero if the character is in the set of alphabetic characters. Otherwise, it is zero.

### Comments

The language driver for the current language (the language the user selected at setup or by using Control Panel) determines whether the character is in the set. If no language has been set, Windows uses an internal function.

### Example

The following example uses the IsCharAlpha function to find the first nonalphabetic character in a string:

```
for (lpzNon = lpz; IsCharAlpha(*lpzNon);
     lpzNon = AnsiNext(lpzNon));
```

### See Also

IsCharAlphaNumeric

## IsCharAlphaNumeric (3.0)

```
BOOL IsCharAlphaNumeric(chTest)
char chTest;      /* character to test */
```

The IsCharAlphaNumeric function determines whether a character is in the set of language-defined alphabetic or numeric characters.

| Parameter | Description                           |
|-----------|---------------------------------------|
| chTest    | Specifies the character to be tested. |

### Returns

The return value is nonzero if the character is in either the set of alphabetic characters or the set of numeric characters. Otherwise, it is zero.

### Comments

The language driver for the current language (the language the user selected at setup or by using Control Panel) determines whether the character is in the set. If no language driver is selected, Windows uses an internal function.

### Example

The following example uses the IsCharAlphaNumeric function to find the first nonalphanumeric character in a string:

```
for (lpzNon = lpz; IsCharAlphaNumeric(*lpzNon);
     lpzNon = AnsiNext(lpzNon));
```

### See Also

IsCharAlpha

## IsCharLower (3.0)

```
BOOL IsCharLower(chTest)
char chTest;      /* character to test */
```

The IsCharLower function determines whether a character is in the set of language-defined lowercase characters.

| Parameter | Description                           |
|-----------|---------------------------------------|
| chTest    | Specifies the character to be tested. |

### Returns

The return value is nonzero if the character is lowercase. Otherwise, it is zero.

### Comments

The language driver for the current language (the language selected at setup or by using Control Panel) determines whether the character is in the set. If no language driver is selected, Windows uses an internal function.

### Example

The following example uses the IsCharLower function to find the first lowercase character in a string:

```
/* Look through string for a lowercase character. */

for (lpszLower = lpsz;
     !IsCharLower(*lpszLower) && lpszLower != '\0';
     lpszLower = AnsiNext(lpszLower));

/* Return NULL if no lowercase character is found. */

if (lpszLower == '\0')
    lpszLower = NULL;
```

### See Also

IsCharUpper

## IsCharUpper (3.0)

```
BOOL IsCharUpper(chTest)
char chTest;      /* character to test */
```

The IsCharUpper function determines whether a character is in the set of language-defined uppercase characters.

| Parameter | Description                           |
|-----------|---------------------------------------|
| chTest    | Specifies the character to be tested. |

### Returns

The return value is nonzero if the character is uppercase. Otherwise, it is zero.

### Comments

The language driver for the current language (the language the user selected at setup or by using Control Panel) determines whether the character is in the set. If no language driver is selected, Windows uses an internal function.

### Example

The following example uses the IsCharUpper function to find the first uppercase character in a string:

```
/* Look through the string for an uppercase character. */

for (lpszUpper = lpsz;
     !IsCharUpper(*lpszUpper) && lpszUpper != '\0';
     lpszUpper = AnsiNext(lpszUpper));

/* Return NULL if no uppercase character is found. */

if (lpszUpper == '\0')
    lpszUpper = NULL;
```

### See Also

IsCharLower

## IsChild (2.x)

```
BOOL IsChild(hwndParent, hwndChild)
HWND hwndParent;      /* handle of parent window */
HWND hwndChild;      /* handle of child window */
/
```

The IsChild function tests whether a given window is a child or other direct descendant of a given parent window. A child window is the direct descendant of a given parent window if that parent window is in the chain of parent windows leading from the original pop-up window to the child window.

| Parameter  | Description                               |
|------------|-------------------------------------------|
| hwndParent | Identifies the parent window.             |
| hwndChild  | Identifies the child window to be tested. |

### Returns

The return value is nonzero if the child window is a descendant of the parent window. Otherwise, it is zero.

### See Also

[SetParent](#)



## IsClipboardFormatAvailable (2.x)

```
BOOL IsClipboardFormatAvailable(uFormat)
UINT uFormat;          /* registered clipboard format */
```

The IsClipboardFormatAvailable function specifies whether data of a certain format exists on the clipboard.

| Parameter | Description                                                                                                                             |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| uFormat   | Specifies a registered clipboard format. For information about clipboard formats, see the description of the SetClipboardData function. |

### Returns

The return value is nonzero if data of the specified format is on the clipboard. Otherwise, the return value is zero.

### Comments

This function is typically called during processing of the WM\_INITMENU or WM\_INITMENUPOPUP message to determine whether the clipboard contains data that the application can paste. If such data is present, the application typically enables the Paste command (in its Edit menu).

### See Also

CountClipboardFormats, EnumClipboardFormats, GetClipboardFormatName, GetPriorityClipboardFormat, RegisterClipboardFormat, SetClipboardData, WM\_INITMENU, WM\_INITMENUPOPUP

## IsDialogMessage (2.x)

```
BOOL IsDialogMessage(hwndDlg, lpmsg)
HWND hwndDlg;          /* handle of dialog box          */
MSG FAR* lpmsg;        /* address of structure with message */
/
```

The IsDialogMessage function determines whether the specified message is intended for the given modeless dialog box and, if it is, processes the message.

| Parameter | Description                                                                |
|-----------|----------------------------------------------------------------------------|
| hwndDlg   | Identifies the dialog box.                                                 |
| lpmsg     | Points to an <u>MSG</u> structure that contains the message to be checked. |

### Returns

The return value is nonzero if the message has been processed. Otherwise, it is zero.

### Comments

Although IsDialogMessage is intended for modeless dialog boxes, it can be used with any window that contains controls, enabling such windows to provide the same keyboard selection as in a dialog box.

When IsDialogMessage processes a message, it checks for keyboard messages and converts them into selection commands for the corresponding dialog box. For example, the TAB key, when pressed, selects the next control or group of controls, and the DOWN ARROW key, when pressed, selects the next control in a group.

If a message is processed by IsDialogMessage, it must not be passed to the TranslateMessage or DispatchMessage function. This is because IsDialogMessage performs all necessary translating and dispatching of messages.

IsDialogMessage sends WM\_GETDLGCODE messages to the dialog box procedure to determine which keys should be processed.

IsDialogMessage can send DM\_GETDEFID and DM\_SETDEFID messages to the window. These messages are defined in the WINDOWS.H header file as WM\_USER and WM\_USER+1, so conflicts are possible with application-defined messages having the same values.

### See Also

DispatchMessage, SendDlgItemMessage, TranslateMessage, MSG, WM\_GETDLGCODE

## IsDlgButtonChecked (2.x)

```
UINT IsDlgButtonChecked(HWND hDlg, int idButton)
/* handle of dialog box */
int idButton; /* button identifier */
```

The `IsDlgButtonChecked` function determines whether a button has a check mark next to it and whether a three-state button is grayed, checked, or neither.

| Parameter             | Description                                         |
|-----------------------|-----------------------------------------------------|
| <code>hDlg</code>     | Identifies the dialog box that contains the button. |
| <code>idButton</code> | Specifies the identifier of the button.             |

### Returns

The return value is nonzero if the specified button is checked, 0 if it is not, or -1 if the `hDlg` parameter is invalid. For three-state buttons, the return value is 2 if the button is grayed, 1 if the button is checked, 0 if it is unchecked, or -1 if `hDlg` is invalid.

### Comments

The `IsDlgButtonChecked` function sends a `BM_GETCHECK` message to the button.

### See Also

`CheckDlgButton`, `CheckRadioButton`, `BM_GETCHECK`

## IsIconic (2.x)

```
BOOL IsIconic(HWND hwnd) /* handle of window */
```

The IsIconic function determines whether the given window is minimized (iconic).

| Parameter | Description            |
|-----------|------------------------|
| hwnd      | Identifies the window. |

### Returns

The return value is nonzero if the window is minimized. Otherwise, it is zero.

### See Also

CloseWindow, IsZoomed

## IsMenu (3.1)

```
BOOL IsMenu(hmenu)
HMENU hmenu;    /* handle of menu */
```

The IsMenu function determines whether the given handle is a menu handle.

| Parameter | Description                         |
|-----------|-------------------------------------|
| hmenu     | Identifies the handle to be tested. |

### Returns

The return value is zero if the handle is definitely not a menu handle. A nonzero return value does not guarantee that the handle is a menu handle, however; for nonzero return values, the application should conduct further tests to verify the handle.

### Comments

An application should use this function only to ensure that a given handle is not a menu handle.

### See Also

[CreateMenu](#), [CreatePopupMenu](#), [DestroyMenu](#), [GetMenu](#)

## IsRectEmpty (2.x)

```
BOOL IsRectEmpty(lprc)
const RECT FAR* lprc;          /* address of structure with rectangle */
```

The IsRectEmpty function determines whether the specified rectangle is empty. A rectangle is empty if its width or height is zero, or if both are zero.

| Parameter | Description                                                                       |
|-----------|-----------------------------------------------------------------------------------|
| lprc      | Points to a <u>RECT</u> structure that contains the coordinates of the rectangle. |

### Returns

The return value is nonzero if the rectangle is empty. Otherwise, it is zero.

### Example

The following example uses the IsRectEmpty function to determine whether a rectangle is empty and then displays a message box giving the status of the rectangle:

```
RECT rc;

if (IsRectEmpty((LPRECT) &rc))
    MessageBox(hwnd, "Rectangle is empty.",
    "Rectangle Status", MB_OK);
else
    MessageBox(hwnd, "Rectangle is not empty.",
    "Rectangle Status", MB_OK);
```

See Also

RECT

## IsWindow (2.x)

```
BOOL IsWindow(hwnd)  
HWND hwnd;    /* handle of window */
```

The IsWindow function determines whether the given window handle is valid.

| Parameter | Description          |
|-----------|----------------------|
| hwnd      | Identifies a window. |

### Returns

The return value is nonzero if the window handle is valid. Otherwise, it is zero.

### See Also

[IsWindowEnabled](#), [IsWindowVisible](#)

## IsWindowEnabled (2.x)

```
BOOL IsWindowEnabled(hwnd)
HWND hwnd;    /* handle of window to test */
```

The IsWindowEnabled function determines whether the given window is enabled for mouse and keyboard input.

| Parameter | Description            |
|-----------|------------------------|
| hwnd      | Identifies the window. |

### Returns

The return value is nonzero if the window is enabled. Otherwise, it is zero.

### Comments

A child window receives input only if it is both enabled and visible.

### See Also

[EnableWindow](#), [IsWindowVisible](#)



## IsWindowVisible (2.x)

```
BOOL IsWindowVisible(HWND)
HWND hwnd; /* handle of window to test */
```

The IsWindowVisible function determines the visibility state of the given window.

| Parameter | Description            |
|-----------|------------------------|
| hwnd      | Identifies the window. |

### Returns

The return value is nonzero if the specified window is visible on the screen (has the WS\_VISIBLE style bit set). The return value is zero if the window is not visible. Because the return value reflects the value of the window's WS\_VISIBLE flag, it may be nonzero even if the window is totally obscured by other windows.

### Comments

A window possesses a visibility state indicated by the WS\_VISIBLE style bit. When this style bit is set, the window is displayed and subsequent drawing into the window is displayed as long as the window has the style bit set.

Any drawing to a window that has the WS\_VISIBLE style will not be displayed if the window is covered by other windows or is clipped by its parent window.

### See Also

ShowWindow

## IsZoomed (2.x)

```
BOOL IsZoomed(hwnd)
HWND hwnd; /* handle of window */
```

The IsZoomed function determines whether the given window is maximized.

| Parameter | Description            |
|-----------|------------------------|
| hwnd      | Identifies the window. |

### Returns

The return value is nonzero if the window is maximized. Otherwise, it is zero.

### See Also

[IsIconic](#)

## KillTimer (2.x)

```
BOOL KillTimer(HWND hwnd, UINT idTimer)
/* handle of window that installed timer */
/* timer */
/* identifier */
/
```

The KillTimer function removes the specified timer. Any pending WM\_TIMER messages associated with the timer are removed from the message queue.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                            |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window associated with the timer to be removed. This must be the same value passed as the hwnd parameter of the <u>SetTimer</u> function that created the timer.                                                                                                                                                                        |
| idTimer   | Identifies the timer to be removed. If the application called <u>SetTimer</u> with the hwnd parameter set to NULL, this parameter must be the timer identifier returned by <u>SetTimer</u> . If the hwnd parameter of <u>SetTimer</u> was a valid window handle, this parameter must be the value of the idTimer parameter passed to <u>SetTimer</u> . |

### Returns

The return value is nonzero if the function is successful. It is zero if the KillTimer function could not find the specified timer.

### See Also

SetTimer, WM\_TIMER

## LoadAccelerators (2.x)

```
HACCEL LoadAccelerators(hinst, lpszTableName)
HINSTANCE hinst;          /* handle of module to load from */
LPCSTR lpszTableName;    /* address of table name */
/
```

The LoadAccelerators function loads the specified accelerator table.

| Parameter     | Description                                                                                             |
|---------------|---------------------------------------------------------------------------------------------------------|
| hinst         | Identifies an instance of the module whose executable file contains the accelerator table to be loaded. |
| lpszTableName | Points to a null-terminated string that names the accelerator table to be loaded.                       |

### Returns

The return value is the handle of the loaded accelerator table if the function is successful. Otherwise, it is NULL.

### Comments

If the accelerator table has not yet been loaded, the function loads it from the given executable file.

Accelerator tables loaded from resources are freed automatically when the application terminates.

■ **LoadBitmap (2.x)**

```

HBITMAP LoadBitmap(hinst, lpszBitmap)
HINSTANCE hinst;          /* handle of application instance */
LPCSTR lpszBitmap;      /*
address of bitmap name
/

```

The LoadBitmap function loads the specified bitmap resource from the given module's executable file.

| Parameter  | Description                                                                                                                                                                                                                                                                                   |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst      | Identifies the instance of the module whose executable file contains the bitmap to be loaded.                                                                                                                                                                                                 |
| lpszBitmap | Points to a null-terminated string that contains the name of the bitmap resource to be loaded. Alternatively, this parameter can consist of the resource identifier in the low-order word and zero in the high-order word. The <u>MAKEINTRESOURCE</u> macro can be used to create this value. |

**Returns**

The return value is the handle of the specified bitmap if the function is successful. Otherwise, it is NULL.

**Comments**

If the bitmap pointed to by lpszBitmap does not exist or if there is insufficient memory to load the bitmap, the function fails.

The application must call the DeleteObject function to delete each bitmap handle returned by the LoadBitmap function. This also applies to the following predefined bitmaps.

An application can use the LoadBitmap function to access the predefined bitmaps used by Windows. To do so, the application must set the hinst parameter to NULL and the lpszBitmap parameter to one of the following values:

|                 |                 |
|-----------------|-----------------|
| OBM_BTNCORNERS  | OBM_OLD_RESTORE |
| OBM_BTFSIZE     | OBM_OLD_RGARROW |
| OBM_CHECK       | OBM_OLD_UPARROW |
| OBM_CHECKBOXES  | OBM_OLD_ZOOM    |
| OBM_CLOSE       | OBM_REDUCE      |
| OBM_COMBO       | OBM_REDUCED     |
| OBM_DNARROW     | OBM_RESTORE     |
| OBM_DNARROWD    | OBM_RESTORED    |
| OBM_DNARROWI    | OBM_RGARROW     |
| OBM_LFARROW     | OBM_RGARROWD    |
| OBM_LFARROWD    | OBM_RGARROWI    |
| OBM_LFARROWI    | OBM_SIZE        |
| OBM_MNARROW     | OBM_UPARROW     |
| OBM_OLD_CLOSE   | OBM_UPARROWD    |
| OBM_OLD_DNARROW | OBM_UPARROWI    |
| OBM_OLD_LFARROW | OBM_ZOOM        |
| OBM_OLD_REDUCE  | OBM_ZOOMD       |

Bitmap names that begin with OBM\_OLD represent bitmaps used by Windows versions earlier than 3.0.

The bitmaps identified by OBM\_DNARROWI, OBM\_LFARROWI, OBM\_RGARROWI, and OBM\_UPARROWI are new for Windows 3.1. These bitmaps are not found in device drivers for previous versions of Windows.

Note that for an application to use any of the OBM\_ constants, the constant OEMRESOURCE must be defined before the WINDOWS.H header file is included.

The following shows the appearance of each of the OBM\_ bitmaps.

See Also  
DeleteObject

Windows 3.1 changes

The following bitmaps have been added:

OBM\_UPARROWI  
OBM\_DNARROWI  
OBM\_RGARROWI  
OBM\_LFARROWI

## LoadCursor (2.x)

```
HCURSOR LoadCursor(hinst, pszCursor)
HINSTANCE hinst;          /* handle of application instance */
LPCSTR pszCursor;
/* cursor-name string or cursor resource identifier
/
```

The LoadCursor function loads the specified cursor resource from the executable file associated with the given application instance.

| Parameter | Description                                                                                                                                                                                                                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst     | Identifies an instance of the module whose executable file contains the cursor to be loaded.                                                                                                                                                                                                  |
| pszCursor | Points to a null-terminated string that contains the name of the cursor resource to be loaded. Alternatively, this parameter can consist of the resource identifier in the low-order word and zero in the high-order word. The <u>MAKEINTRESOURCE</u> macro can be used to create this value. |

### Returns

The return value is the handle of the newly loaded cursor if the function is successful. Otherwise, it is NULL.

### Comments

The function loads the cursor resource only if it has not been loaded; otherwise, it retrieves a handle of the existing resource. The LoadCursor function returns a valid cursor handle only if the pszCursor parameter points to a cursor resource. If pszCursor points to any type of resource other than a cursor (such as an icon), the return value will not be NULL, even though it is not a valid cursor handle.

An application can use the LoadCursor function to access the predefined cursors used by Windows. To do this, the application must set the hinst parameter to NULL and the pszCursor parameter to one of the following values:

| Value               | Meaning                                                             |
|---------------------|---------------------------------------------------------------------|
| <u>IDC_ARROW</u>    | Standard arrow cursor.                                              |
| <u>IDC_CROSS</u>    | Crosshair cursor.                                                   |
| <u>IDC_IBEAM</u>    | Text I-beam cursor.                                                 |
| <u>IDC_ICON</u>     | Empty icon.                                                         |
| <u>IDC_SIZE</u>     | A square with a smaller square inside its lower-right corner.       |
| <u>IDC_SIZENESW</u> | Double-pointed cursor with arrows pointing northeast and southwest. |
| <u>IDC_SIZENS</u>   | Double-pointed cursor with arrows pointing north and south.         |
| <u>IDC_SIZENWSE</u> | Double-pointed cursor with arrows pointing northwest and southeast. |
| <u>IDC_SIZEWE</u>   | Double-pointed cursor with arrows pointing west and east.           |
| <u>IDC_UPARROW</u>  | Vertical arrow cursor.                                              |
| <u>IDC_WAIT</u>     | Hourglass cursor.                                                   |

It is not necessary to destroy these system cursors. An application should use the DestroyCursor function to destroy any private cursors it loads.

### See Also

DestroyCursor, SetCursor, ShowCursor, MAKEINTRESOURCE



IDC\_ARROW MAKEINTRESOURCE(32512)

Standard arrow cursor.

IDC\_ARROW MAKEINTRESOURCE(32512)

IDC\_CROSS MAKEINTRESOURCE(32515)

Crosshair cursor.

IDC\_CROSS MAKEINTRESOURCE(32515)

IDC\_IBEAM MAKEINTRESOURCE(32513)

Text I-beam cursor.

IDC\_IBEAM MAKEINTRESOURCE(32513)

IDC\_ICON MAKEINTRESOURCE(32641)  
Empty icon.

IDC\_ICON MAKEINTRESOURCE(32641)



IDC\_SIZE MAKEINTRESOURCE(32640)

A square with a smaller square inside its lower-right corner.

IDC\_SIZE MAKEINTRESOURCE(32640)

IDC\_SIZENESW MAKEINTRESOURCE(32643)

Double-pointed cursor with arrows pointing northeast and southwest.

IDC\_SIZENESW MAKEINTRESOURCE(32643)

IDC\_SIZENS MAKEINTRESOURCE(32645)

Double-pointed cursor with arrows pointing north and south.

IDC\_SIZENS MAKEINTRESOURCE(32645)

IDC\_SIZENWSE MAKEINTRESOURCE(32642)

Double-pointed cursor with arrows pointing northwest and southeast.

IDC\_SIZENWSE MAKEINTRESOURCE(32642)



IDC\_SIZEWE MAKEINTRESOURCE(32644)

Double-pointed cursor with arrows pointing west and east.

IDC\_SIZEWE MAKEINTRESOURCE(32644)

IDC\_UPARROW MAKEINTRESOURCE(32516)

Vertical arrow cursor.

IDC\_UPARROW MAKEINTRESOURCE(32516)

IDC\_WAIT MAKEINTRESOURCE(32514)  
Hourglass cursor.

IDC\_WAIT MAKEINTRESOURCE(32514)

## LoadIcon (2.x)

HICON LoadIcon(hinst, pszIcon)

HINSTANCE hinst; /\* handle of application instance \*/

LPCSTR pszIcon; /

\* icon-name string or icon resource identifier

/

The LoadIcon function loads the specified icon resource from the executable file associated with the given application instance.

| Parameter | Description                                                                                                                                                                                                                                                                                 |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst     | Identifies an instance of the module whose executable file contains the icon to be loaded. This parameter must be NULL when a system icon is being loaded.                                                                                                                                  |
| pszIcon   | Points to a null-terminated string that contains the name of the icon resource to be loaded. Alternatively, this parameter can consist of the resource identifier in the low-order word and zero in the high-order word. The <u>MAKEINTRESOURCE</u> macro can be used to create this value. |

### Returns

The return value is the handle of the newly loaded icon if the function is successful. Otherwise, it is NULL.

### Comments

This function loads the icon resource only if it has not been loaded; otherwise, it retrieves a handle of the existing resource.

An application can use the LoadIcon function to access the predefined icons used by Windows. To do this, the application must set the hinst parameter to NULL and the pszIcon parameter to one of the following values:

| Value                  | Meaning                                              |
|------------------------|------------------------------------------------------|
| <u>IDI_APPLICATION</u> | Default application icon.                            |
| <u>IDI_ASTERISK</u>    | Asterisk (used in informative messages).             |
| <u>IDI_EXCLAMATION</u> | Exclamation point (used in warning messages).        |
| <u>IDI_HAND</u>        | Hand-shaped icon (used in serious warning messages). |
| <u>IDI_QUESTION</u>    | Question mark (used in prompting messages).          |

It is not necessary to destroy these system icons. An application should use the DestroyIcon function to destroy any private icons it loads.

The following shows all of the system icons.

sysico

See Also

DestroyIcon, DrawIcon, MAKEINTRESOURCE

IDI\_APPLICATION MAKEINTRESOURCE(32512)

Default application icon.



IDI\_APPLICATION MAKEINTRESOURCE(32512)

IDI\_ASTERISK MAKEINTRESOURCE(32516)

Asterisk (used in informative messages).

IDI\_ASTERISK MAKEINTRESOURCE(32516)

IDI\_EXCLAMATION MAKEINTRESOURCE(32515)

Exclamation point (used in warning messages).

IDI\_EXCLAMATION MAKEINTRESOURCE(32515)

IDI\_HAND MAKEINTRESOURCE(32513)

Hand-shaped icon (used in serious warning messages).

IDI\_HAND MAKEINTRESOURCE(32513)

IDI\_QUESTION MAKEINTRESOURCE(32514)

Question mark (used in prompting messages).



IDI\_QUESTION MAKEINTRESOURCE(32514)

## LoadMenu (2.x)

```
HMENU LoadMenu(hinst, lpszMenuName)
HINSTANCE hinst;          /* handle of application instance */
LPCSTR lpszMenuName;
/* menu-name string or menu resource identifier
/
```

The LoadMenu function loads the specified menu resource from the executable file associated with the given application instance.

| Parameter    | Description                                                                                                                                                                                                                                                                                 |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst        | Identifies an instance of the module whose executable file contains the menu to be loaded.                                                                                                                                                                                                  |
| lpszMenuName | Points to a null-terminated string that contains the name of the menu resource to be loaded. Alternatively, this parameter can consist of the resource identifier in the low-order word and zero in the high-order word. The <u>MAKEINTRESOURCE</u> macro can be used to create this value. |

### Returns

The return value is the handle of the menu resource if the function is successful. Otherwise, it is NULL.

### Comments

Before exiting, an application must free system resources associated with a menu if the menu is not assigned to a window. An application frees a menu by calling the DestroyMenu function.

### Example

The following example loads a menu resource, and then assigns the menu to a window:

```
HMENU hmenu;
```

```
hmenu = LoadMenu(hinst, "ColorMenu");
```

```
SetMenu(hwnd, hmenu);
```

### See Also

DestroyMenu, LoadMenuIndirect, SetMenu, MAKEINTRESOURCE

## LoadMenuIndirect (2.x)

```
HMENU LoadMenuIndirect(lpmith)
const void FAR* lpmith;          /* address of menu template */
```

The LoadMenuIndirect function loads the specified menu template in memory. A menu template is a header followed by a collection of one or more MENUIITEMTEMPLATE structures, each of which may contain one or more menu items and pop-up menus.

| Parameter | Description                                                                                                                                                                                                                                                            |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpmith    | Points to a menu template, which consists of a menu-template header and one or more menu item templates. The menu template header consists of a <u>MENUIITEMTEMPLATEHEADER</u> structure.<br>Each menu item template consists of a <u>MENUIITEMTEMPLATE</u> structure. |

### Returns

The return value is the handle of a menu if the function is successful. Otherwise, it is NULL.

### Comments

Before exiting, an application must free system resources associated with a menu if the menu is not assigned to a window. An application frees a menu by calling the DestroyMenu function.

### Example

The following example retrieves a menu handle for a menu template resource that has been loaded into memory, gives the menu handle to a window, and then unlocks and frees the resource:

```
HRSRC hrsrcResInfo;
HGLOBAL hglbResMenu;
char FAR* lpResMenu;
HMENU hmenu;

case IDM_NEWMENU:
    hrsrcResInfo = FindResource(hinst, "DynaMenu", RT_MENU);
    hglbResMenu = LoadResource(hinst, hrsrcResInfo);
    lpResMenu = LockResource(hglbResMenu);
    hmenu = LoadMenuIndirect(lpResMenu);

    DestroyMenu(GetMenu(hwnd));
    SetMenu(hwnd, hmenu);

    UnlockResource(hglbResMenu);
    FreeResource(hglbResMenu);

    break;
```

### See Also

DestroyMenu, LoadMenu, SetMenu, MENUIITEMTEMPLATE

## LoadString (2.x)

```
int LoadString(hinst, idResource, lpszBuffer, cbBuffer)
HINSTANCE hinst;          /* handle of module containing string resource */
UINT idResource;        /* resource identifier */
LPSTR lpszBuffer;       /* address of buffer for resource */
int cbBuffer;           /* size of buffer */
```

The LoadString function loads the specified string resource.

| Parameter  | Description                                                                                                                                                                                           |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hinst      | Identifies an instance of the module whose executable file contains the string resource to be loaded.                                                                                                 |
| idResource | Specifies the integer identifier of the string to be loaded.                                                                                                                                          |
| lpszBuffer | Points to the buffer that will receive the null-terminated string.                                                                                                                                    |
| cbBuffer   | Specifies the buffer size, in bytes. The buffer should be large enough for the string and its terminating null character. The string is truncated if it is longer than the number of bytes specified. |

### Returns

The return value specifies the number of bytes copied into the buffer, if the function is successful. It is zero if the string resource does not exist.

### LockInput (3.1)

```
BOOL LockInput(hReserved, hwndInput, fLock)
HANDLE hReserved;          /* reserved, must be NULL          */
HWND hwndInput;           /* handle of window to receive all input */
/* the lock/unlock flag
/
```

The LockInput function locks input to all tasks except the current one, if the fLock parameter is TRUE. The given window is made system modal; that is, it will receive all input. If fLock is FALSE, LockInput unlocks input and restores the system to its unlocked state.

| Parameter | Description                                                                                                                                    |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------|
| hReserved | This parameter is reserved and must be NULL.                                                                                                   |
| hwndInput | Identifies the window that is to receive all input. This window must be in the current task. If fLock is FALSE, this parameter should be NULL. |
| fLock     | Indicates whether to lock or unlock input. A value of TRUE locks input; a value of FALSE unlocks input.                                        |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

Before entering hard mode, a Windows-based debugger calls LockInput, specifying TRUE for the fLock parameter. This action saves the current global state. To exit hard mode, the debugger calls LockInput, specifying FALSE for fLock. This restores the global state to the conditions that existed when the debugger entered hard mode. A debugger must restore the global state before exiting. Calls to LockInput cannot be nested.

#### See Also

[DirectedYield](#)

## LockWindowUpdate (3.1)

```
BOOL LockWindowUpdate(hwndLock)
HWND hwndLock; /* handle of window */
```

The LockWindowUpdate function disables or reenables drawing in the given window. A locked window cannot be moved. Only one window can be locked at a time.

| Parameter | Description                                                                                                                  |
|-----------|------------------------------------------------------------------------------------------------------------------------------|
| hwndLock  | Identifies the window in which drawing will be disabled. If this parameter is NULL, drawing in the locked window is enabled. |

### Returns

The return value is nonzero if the function is successful. It is zero if a failure occurs or if the LockWindowUpdate function has been used to lock another window.

### Comments

If an application with a locked window (or any locked child windows) calls the GetDC, GetDCEX, or BeginPaint function, the called function returns a device context whose visible region is empty. This will occur until the application unlocks the window by calling LockWindowUpdate, specifying a value of NULL for hwndLock.

While window updates are locked, the system keeps track of the bounding rectangle of any drawing operations to device contexts associated with a locked window. When drawing is reenabled, this bounding rectangle is invalidated in the locked window and its child windows to force an eventual WM\_PAINT message to update the screen. If no drawing has occurred while the window updates were locked, no area is invalidated.

The LockWindowUpdate function does not make the given window invisible and does not clear the WS\_VISIBLE style bit.

## lstrcmp (3.0)

```
int lstrcmp(lpszString1, lpszString2)
LPCSTR lpszString1;      /* address of first string */
LPCSTR lpszString2;      /* address of
second string */
/
```

The lstrcmp function compares two character strings. The comparison is case-sensitive.

| Parameter   | Description                                                 |
|-------------|-------------------------------------------------------------|
| lpszString1 | Points to the first null-terminated string to be compared.  |
| lpszString2 | Points to the second null-terminated string to be compared. |

### Returns

The return value is less than zero if the string specified in lpszString1 is less than the string specified in lpszString2, is greater than zero if lpszString1 is greater than lpszString2, and is zero if the two strings are equal.

### Comments

The lstrcmp function compares two strings by checking the first characters against each other, the second characters against each other, and so on, until it finds an inequality or reaches the ends of the strings. The function returns the difference of the values of the first unequal characters it encounters. For example, lstrcmp determines that "abcz" is greater than "abcdefg" and returns the difference of "z" and "d".

The language driver for the language selected by the user determines which string is greater (or whether the strings are the same). If no language driver is selected, Windows uses an internal function. With the Windows United States language functions, uppercase characters have lower values than lowercase characters.

With a double-byte character set (DBCS) version of Windows, this function can compare two DBCS strings.

Both strings must be less than 64K in size.

### See Also

[lstrcmpi](#)

## lstrcmpi (3.0)

```
int lstrcmpi(lpszString1, lpszString2)
LPCSTR lpszString1;      /* address of first string */
LPCSTR lpszString2;      /* address of
second string */
/
```

The lstrcmpi function compares the two strings. The comparison is not case-sensitive.

| Parameter   | Description                                                 |
|-------------|-------------------------------------------------------------|
| lpszString1 | Points to the first null-terminated string to be compared.  |
| lpszString2 | Points to the second null-terminated string to be compared. |

### Returns

The return value is less than zero if the string specified in lpszString1 is less than the string specified in lpszString2, is greater than zero if lpszString1 is greater than lpszString2, and is zero if the two strings are equal.

### Comments

The lstrcmpi function compares two strings by checking the first characters against each other, the second characters against each other, and so on, until it finds an inequality or reaches the ends of the strings. The function returns the difference of the values of the first unequal characters it encounters. For example, lstrcmpi determines that "abcz" is greater than "abcdefg" and returns the difference of "z" and "d".

The language driver for the language selected by the user determines which string is greater (or whether the strings are the same). If no language driver is selected, Windows uses an internal function.

With a double-byte character set (DBCS) version of Windows, this function can compare two DBCS strings.

Both strings must be less than 64K in size.

### See Also

[lstrcmp](#)



## MapDialogRect (2.x)

```
void MapDialogRect(HWND hDlg, LPRECT FAR* lpRect)
/* handle of dialog box */
/* address of
structure with rectangle
/
```

The MapDialogRect function converts (maps) the specified dialog box units to screen units (pixels).

| Parameter | Description                                                                                                                                      |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| hDlg      | Identifies a dialog box. This dialog box must have been created by using the <a href="#">CreateDialog</a> or <a href="#">DialogBox</a> function. |
| lpRect    | Points to a <a href="#">RECT</a> structure that contains the dialog box coordinates to be converted.                                             |

### Returns

This function does not return a value.

### Comments

The MapDialogRect function converts the dialog box units of a rectangle to screen units. Dialog box units are defined in terms of the current dialog base unit, which is derived from the average width and height of characters in the font used for dialog box text. Typically, dialog boxes use the System font, but an application can specify a different font by using the DS\_SETFONT style in the resource-definition file.

One horizontal unit is one-fourth of the dialog box base width unit, and one vertical unit is one-eighth of the dialog box base height unit. The [GetDialogBaseUnits](#) function retrieves the dialog box base units in pixels.

### See Also

[CreateDialog](#), [DialogBox](#), [GetDialogBaseUnits](#), [RECT](#)

## MessageBeep (2.x)

```
void MessageBeep(uAlert)
UINT uAlert;      /* alert level */
```

The MessageBeep function plays a waveform sound corresponding to a given system alert level. The sound for each alert level is identified by an entry in the [sounds] section of the WIN.INI initialization file.

| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |    |                                                               |                 |                                                                                            |                    |                                                                                               |             |                                                                                        |                 |                                                                                            |       |                                                                                           |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|----|---------------------------------------------------------------|-----------------|--------------------------------------------------------------------------------------------|--------------------|-----------------------------------------------------------------------------------------------|-------------|----------------------------------------------------------------------------------------|-----------------|--------------------------------------------------------------------------------------------|-------|-------------------------------------------------------------------------------------------|
| uAlert             | Specifies the alert level. This parameter can be one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |    |                                                               |                 |                                                                                            |                    |                                                                                               |             |                                                                                        |                 |                                                                                            |       |                                                                                           |
|                    | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>-1</td><td>Produces a standard beep sound by using the computer speaker.</td></tr><tr><td>MB_ICONASTERISK</td><td>Plays the sound identified by the SystemAsterisk entry in the [sounds] section of WIN.INI.</td></tr><tr><td>MB_ICONEXCLAMATION</td><td>Plays the sound identified by the SystemExclamation entry in the [sounds] section of WIN.INI.</td></tr><tr><td>MB_ICONHAND</td><td>Plays the sound identified by the SystemHand entry in the [sounds] section of WIN.INI.</td></tr><tr><td>MB_ICONQUESTION</td><td>Plays the sound identified by the SystemQuestion entry in the [sounds] section of WIN.INI.</td></tr><tr><td>MB_OK</td><td>Plays the sound identified by the SystemDefault entry in the [sounds] section of WIN.INI.</td></tr></tbody></table> | Value | Meaning | -1 | Produces a standard beep sound by using the computer speaker. | MB_ICONASTERISK | Plays the sound identified by the SystemAsterisk entry in the [sounds] section of WIN.INI. | MB_ICONEXCLAMATION | Plays the sound identified by the SystemExclamation entry in the [sounds] section of WIN.INI. | MB_ICONHAND | Plays the sound identified by the SystemHand entry in the [sounds] section of WIN.INI. | MB_ICONQUESTION | Plays the sound identified by the SystemQuestion entry in the [sounds] section of WIN.INI. | MB_OK | Plays the sound identified by the SystemDefault entry in the [sounds] section of WIN.INI. |
| Value              | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |    |                                                               |                 |                                                                                            |                    |                                                                                               |             |                                                                                        |                 |                                                                                            |       |                                                                                           |
| -1                 | Produces a standard beep sound by using the computer speaker.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |    |                                                               |                 |                                                                                            |                    |                                                                                               |             |                                                                                        |                 |                                                                                            |       |                                                                                           |
| MB_ICONASTERISK    | Plays the sound identified by the SystemAsterisk entry in the [sounds] section of WIN.INI.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |    |                                                               |                 |                                                                                            |                    |                                                                                               |             |                                                                                        |                 |                                                                                            |       |                                                                                           |
| MB_ICONEXCLAMATION | Plays the sound identified by the SystemExclamation entry in the [sounds] section of WIN.INI.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |    |                                                               |                 |                                                                                            |                    |                                                                                               |             |                                                                                        |                 |                                                                                            |       |                                                                                           |
| MB_ICONHAND        | Plays the sound identified by the SystemHand entry in the [sounds] section of WIN.INI.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |    |                                                               |                 |                                                                                            |                    |                                                                                               |             |                                                                                        |                 |                                                                                            |       |                                                                                           |
| MB_ICONQUESTION    | Plays the sound identified by the SystemQuestion entry in the [sounds] section of WIN.INI.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |    |                                                               |                 |                                                                                            |                    |                                                                                               |             |                                                                                        |                 |                                                                                            |       |                                                                                           |
| MB_OK              | Plays the sound identified by the SystemDefault entry in the [sounds] section of WIN.INI.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |    |                                                               |                 |                                                                                            |                    |                                                                                               |             |                                                                                        |                 |                                                                                            |       |                                                                                           |

### Returns

This function does not return a value.

### Comments

MessageBeep returns control to the caller after queuing the sound and plays the sound asynchronously.

If it cannot play the specified alert sound, MessageBeep attempts to play the system default sound. If it cannot play the system default sound, the function produces a standard beep sound by using the computer speaker.

The user can disable the warning beep by using the Windows Control Panel application Sounds.

### See Also

[FlashWindow](#), [MessageBox](#)

### Windows 3.1 changes

The MessageBeep function for Windows version 3.0 and earlier did not accept values for the uAlert parameter.

## MessageBox (2.x)

```
int MessageBox(hwndParent, lpszText, lpszTitle, fuStyle)
HWND hwndParent; /* handle of parent window */
LPCSTR lpszText; /* address of text
in message box
/
LPCSTR lpszTitle;
* address of title of message box
/
UINT fuStyle;
* style of message box
/
```

The MessageBox function creates, displays, and operates a message-box window. The message box contains an application-defined message and title, plus any combination of the predefined icons and push buttons described in the fuStyle parameter.

| Parameter                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|----------------------------|------------------------------------------------------------------------|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|-----------------------------------|----------------------|----------------------------------|------------------------|-------------------------------------|---------------------------|-------------------------------------------------------|--------------------|------------------------------|---------------------------|--------------------------------------------------------------------------------------|------------------------|--------------------------------------------------|--------------------|----------------------------------------------|--------------|-----------------------------------------------|--------------------|-----------------------------------------------------------|-----------------------|--------------------------------------------------------------|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwndParent                 | Identifies the parent window of the message box to be created. If this parameter is NULL, the message box will have no parent window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| lpszText                   | Points to a null-terminated string containing the message to be displayed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| lpszTitle                  | Points to a null-terminated string to be used for the dialog box title. If this parameter is NULL, the default title Error is used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| fuStyle                    | Specifies the contents and behavior of the dialog box. This parameter can be a combination of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
|                            | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>MB_ABORTRETRYIGNORE</u></td><td>The message box contains three push buttons: Abort, Retry, and Ignore.</td></tr><tr><td><u>MB_APPLMODAL</u></td><td>The user must respond to the message box before continuing work in the window identified by the hwndParent parameter. However, the user can move to the windows of other applications and work in those windows. <u>MB_APPLMODAL</u> is the default if neither <u>MB_SYSTEMMODAL</u> nor <u>MB_TASKMODAL</u> is specified.</td></tr><tr><td><u>MB_DEFBUTTON1</u></td><td>The first button is the default. Note that the first button is always the default unless <u>MB_DEFBUTTON2</u> or <u>MB_DEFBUTTON3</u> is specified.</td></tr><tr><td><u>MB_DEFBUTTON2</u></td><td>The second button is the default.</td></tr><tr><td><u>MB_DEFBUTTON3</u></td><td>The third button is the default.</td></tr><tr><td><u>MB_ICONASTERISK</u></td><td>Same as <u>MB_ICONINFORMATION</u>.</td></tr><tr><td><u>MB_ICONEXCLAMATION</u></td><td>An exclamation-point icon appears in the message box.</td></tr><tr><td><u>MB_ICONHAND</u></td><td>Same as <u>MB_ICONSTOP</u>.</td></tr><tr><td><u>MB_ICONINFORMATION</u></td><td>An icon consisting of a lowercase letter "I" in a circle appears in the message box.</td></tr><tr><td><u>MB_ICONQUESTION</u></td><td>A question-mark icon appears in the message box.</td></tr><tr><td><u>MB_ICONSTOP</u></td><td>A stop-sign icon appears in the message box.</td></tr><tr><td><u>MB_OK</u></td><td>The message box contains one push button: OK.</td></tr><tr><td><u>MB_OKCANCEL</u></td><td>The message box contains two push buttons: OK and Cancel.</td></tr><tr><td><u>MB_RETRYCANCEL</u></td><td>The message box contains two push buttons: Retry and Cancel.</td></tr><tr><td><u>MB_SYSTEMMODAL</u></td><td>All applications are suspended until the user responds to the message box. Unless the application specifies <u>MB_ICONHAND</u>, the message box does not become modal until after it</td></tr></tbody></table> | Value | Meaning | <u>MB_ABORTRETRYIGNORE</u> | The message box contains three push buttons: Abort, Retry, and Ignore. | <u>MB_APPLMODAL</u> | The user must respond to the message box before continuing work in the window identified by the hwndParent parameter. However, the user can move to the windows of other applications and work in those windows. <u>MB_APPLMODAL</u> is the default if neither <u>MB_SYSTEMMODAL</u> nor <u>MB_TASKMODAL</u> is specified. | <u>MB_DEFBUTTON1</u> | The first button is the default. Note that the first button is always the default unless <u>MB_DEFBUTTON2</u> or <u>MB_DEFBUTTON3</u> is specified. | <u>MB_DEFBUTTON2</u> | The second button is the default. | <u>MB_DEFBUTTON3</u> | The third button is the default. | <u>MB_ICONASTERISK</u> | Same as <u>MB_ICONINFORMATION</u> . | <u>MB_ICONEXCLAMATION</u> | An exclamation-point icon appears in the message box. | <u>MB_ICONHAND</u> | Same as <u>MB_ICONSTOP</u> . | <u>MB_ICONINFORMATION</u> | An icon consisting of a lowercase letter "I" in a circle appears in the message box. | <u>MB_ICONQUESTION</u> | A question-mark icon appears in the message box. | <u>MB_ICONSTOP</u> | A stop-sign icon appears in the message box. | <u>MB_OK</u> | The message box contains one push button: OK. | <u>MB_OKCANCEL</u> | The message box contains two push buttons: OK and Cancel. | <u>MB_RETRYCANCEL</u> | The message box contains two push buttons: Retry and Cancel. | <u>MB_SYSTEMMODAL</u> | All applications are suspended until the user responds to the message box. Unless the application specifies <u>MB_ICONHAND</u> , the message box does not become modal until after it |
| Value                      | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_ABORTRETRYIGNORE</u> | The message box contains three push buttons: Abort, Retry, and Ignore.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_APPLMODAL</u>        | The user must respond to the message box before continuing work in the window identified by the hwndParent parameter. However, the user can move to the windows of other applications and work in those windows. <u>MB_APPLMODAL</u> is the default if neither <u>MB_SYSTEMMODAL</u> nor <u>MB_TASKMODAL</u> is specified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_DEFBUTTON1</u>       | The first button is the default. Note that the first button is always the default unless <u>MB_DEFBUTTON2</u> or <u>MB_DEFBUTTON3</u> is specified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_DEFBUTTON2</u>       | The second button is the default.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_DEFBUTTON3</u>       | The third button is the default.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_ICONASTERISK</u>     | Same as <u>MB_ICONINFORMATION</u> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_ICONEXCLAMATION</u>  | An exclamation-point icon appears in the message box.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_ICONHAND</u>         | Same as <u>MB_ICONSTOP</u> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_ICONINFORMATION</u>  | An icon consisting of a lowercase letter "I" in a circle appears in the message box.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_ICONQUESTION</u>     | A question-mark icon appears in the message box.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_ICONSTOP</u>         | A stop-sign icon appears in the message box.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_OK</u>               | The message box contains one push button: OK.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_OKCANCEL</u>         | The message box contains two push buttons: OK and Cancel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_RETRYCANCEL</u>      | The message box contains two push buttons: Retry and Cancel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |
| <u>MB_SYSTEMMODAL</u>      | All applications are suspended until the user responds to the message box. Unless the application specifies <u>MB_ICONHAND</u> , the message box does not become modal until after it                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                            |                                                                        |                     |                                                                                                                                                                                                                                                                                                                            |                      |                                                                                                                                                     |                      |                                   |                      |                                  |                        |                                     |                           |                                                       |                    |                              |                           |                                                                                      |                        |                                                  |                    |                                              |              |                                               |                    |                                                           |                       |                                                              |                       |                                                                                                                                                                                       |

is created; consequently, the parent window and other windows continue to receive messages resulting from its activation. System-modal message boxes are used to notify the user of serious, potentially damaging errors that require immediate attention (for example, running out of memory).

**MB\_TASKMODAL**

Same as **MB\_APPLMODAL** except that all the top-level windows belonging to the current task are disabled if the `hwndParent` parameter is `NULL`. This flag should be used when the calling application or library does not have a window handle available but still needs to prevent input to other windows in the current application without suspending other applications.

**MB\_YESNO**

The message box contains two push buttons: Yes and No.

**MB\_YESNOCANCEL**

The message box contains three push buttons: Yes, No, and Cancel.

**Returns**

The return value is zero if there is not enough memory to create the message box. Otherwise, it is one of the following menu-item values returned by the dialog box:

| Value                  | Meaning                     |
|------------------------|-----------------------------|
| <b><u>IDABORT</u></b>  | Abort button was selected.  |
| <b><u>IDCANCEL</u></b> | Cancel button was selected. |
| <b><u>IDIGNORE</u></b> | Ignore button was selected. |
| <b><u>IDNO</u></b>     | No button was selected.     |
| <b><u>IDOK</u></b>     | OK button was selected.     |
| <b><u>IDRETRY</u></b>  | Retry button was selected.  |
| <b><u>IDYES</u></b>    | Yes button was selected.    |

If a message box has a Cancel button, the **IDCANCEL** value will be returned if either the ESC key is pressed or the Cancel button is selected. If the message box has no Cancel button, pressing ESC has no effect.

**Comments**

When a system-modal message box is created to indicate that the system is low on memory, the strings pointed to by the `lpszText` and `lpszTitle` parameters should not be taken from a resource file, because an attempt to load the resource may fail.

When an application calls the `MessageBox` function and specifies the **MB\_ICONHAND** and **MB\_SYSTEMMODAL** flags for the `fuStyle` parameter, Windows displays the resulting message box regardless of available memory. When these flags are specified, Windows limits the length of the message-box text to three lines. Windows does not automatically break the lines to fit in the message box, however, so the message string must contain carriage returns to break the lines at the appropriate places.

If a message box is created while a dialog box is present, use the handle of the dialog box as the `hwndParent` parameter. The `hwndParent` parameter should not identify a child window, such as a control in a dialog box.

**See Also**  
[Flash Window](#), [MessageBeep](#)

Following are the various system icons that can be used in a message box:

MB\_ABORTRETRYIGNORE 0x0002

The message box contains three push buttons: Abort, Retry, and Ignore.

MB\_ABORTRETRYIGNORE 0x0002

MB\_APPLMODAL 0x0000

The user must respond to the message box before continuing work in the window identified by the `hwndParent` parameter. However, the user can move to the windows of other applications and work in those windows. `MB_APPLMODAL` is the default if neither `MB_SYSTEMMODAL` nor `MB_TASKMODAL` is specified.



MB\_APPLMODAL 0x0000

MB\_DEFBUTTON1 0x0000

The first button is the default. Note that the first button is always the default unless MB\_DEFBUTTON2 or MB\_DEFBUTTON3 is specified.

MB\_DEFBUTTON1 0x0000

MB\_DEFBUTTON2 0x0100

The second button is the default.

MB\_DEFBUTTON2 0x0100

MB\_DEFBUTTON3 0x0200

The third button is the default.

MB\_DEFBUTTON3 0x0200

MB\_ICONASTERISK 0x0040  
Same as MB\_ICONINFORMATION.



MB\_ICONASTERISK 0x0040

MB\_ICONEXCLAMATION 0x0030

An exclamation-point icon appears in the message box.

MB\_ICONEXCLAMATION 0x0030

MB\_ICONHAND 0x0010  
Same as MB\_ICONSTOP.

MB\_ICONHAND 0x0010

MB\_ICONINFORMATION MB\_ICONASTERISK

An icon consisting of a lowercase letter "i" in a circle appears in the message box.

MB\_ICONINFORMATION MB\_ICONASTERISK

MB\_ICONQUESTION 0x0020

A question-mark icon appears in the message box.



MB\_ICONQUESTION 0x0020

MB\_ICONSTOP MB\_ICONHAND

A stop-sign icon appears in the message box.

MB\_ICONSTOP MB\_ICONHAND

MB\_OK 0x0000

The message box contains one push button: OK.

MB\_OK 0x0000

MB\_OKCANCEL 0x0001

The message box contains two push buttons: OK and Cancel.

MB\_OKCANCEL 0x0001

MB\_RETRYCANCEL 0x0005

The message box contains two push buttons: Retry and Cancel.



MB\_RETRYCANCEL 0x0005

MB\_SYSTEMMODAL 0x1000

All applications are suspended until the user responds to the message box. Unless the application specifies MB\_ICONHAND, the message box does not become modal until after it is created; consequently, the parent window and other windows continue to receive messages resulting from its activation. System-modal message boxes are used to notify the user of serious, potentially damaging errors that require immediate attention (for example, running out of memory).

MB\_SYSTEMMODAL 0x1000

MB\_TASKMODAL 0x2000

Same as MB\_APPLMODAL except that all the top-level windows belonging to the current task are disabled if the hwndParent parameter is NULL. This flag should be used when the calling application or library does not have a window handle available but still needs to prevent input to other windows in the current application without suspending other applications.

MB\_TASKMODAL 0x2000

MB\_YESNO 0x0004

The message box contains two push buttons: Yes and No.

MB\_YESNO 0x0004

MB\_YESNOCANCEL 0x0003

The message box contains three push buttons: Yes, No, and Cancel.



MB\_YESNOCANCEL 0x0003

IDABORT 3

Abort button was selected.

IDABORT 3

IDCANCEL 2

Cancel button was selected.

IDCANCEL 2

IDIGNORE 5

Ignore button was selected.

IDIGNORE 5

IDNO 7

No button was selected.



IDNO 7

IDOK 1

OK button was selected.

IDOK 1

IDRETRY 4

Retry button was selected.

IDRETRY 4

IDYES 6

Yes button was selected.

IDYES 6

### MapWindowPoints (3.1)

```
void MapWindowPoints(HWND hwndFrom, HWND hwndTo, LPPOINT lppt, cPoints)
/* handle of window to be mapped from */
/* handle of window to be mapped to */
POINT FAR* lppt;
/* address of structure array with points to map */
UINT cPoints;
/* number of structures in array */
```

The MapWindowPoints function converts (maps) a set of points from a coordinate space relative to one window to a coordinate space relative to another window.

| Parameter | Description                                                                                                                                                                                                     |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwndFrom  | Identifies the window from which points are converted. If this parameter is NULL or HWND_DESKTOP, the points are assumed to be in screen coordinates.                                                           |
| hwndTo    | Identifies the window to which points are converted. If this parameter is NULL or HWND_DESKTOP, the points are converted to screen coordinates.                                                                 |
| lppt      | Points to an array of <u>POINT</u> structures that contain the set of points to be converted. This parameter can also point to a <u>RECT</u> structure, in which case the cPoints parameter should be set to 2. |
| cPoints   | Specifies the number of <u>POINT</u> structures in the array pointed to by the lppt parameter.                                                                                                                  |

#### Returns

This function does not return a value.

#### See Also

ClientToScreen, ScreenToClient



## ModifyMenu (3.0)

```
BOOL ModifyMenu(hmenu, idItem, fuFlags, idNewItem, lpNewItem)
HMENU hmenu;          /* handle of menu */
UINT idItem;          /* menu-item identifier */
/
UINT fuFlags;        /* menu-item flags */
/
UINT idNewItem;     /* new menu-item identifier */
/
LPCSTR lpNewItem;  /* menu-item content */
/
```

The ModifyMenu function changes an existing menu item.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hmenu     | Identifies the menu to change.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| idItem    | Specifies the menu item to change, as determined by the fuFlags parameter. When the fuFlags parameter is <u>MF_BYCOMMAND</u> , the idItem parameter specifies the menu-item identifier. When the fuFlags parameter is <u>MF_BYPOSITION</u> , the idItem parameter specifies the zero-based position of the menu item.                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| fuFlags   | Specifies how the idItem parameter is interpreted and information about the changes to be made to the menu item. It consists of one or more values listed in the following Comments section.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| idNewItem | Specifies either the identifier of the modified menu item or, if fuFlags is set to <u>MF_POPUP</u> , the menu handle of the pop-up menu.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| lpNewItem | Specifies the content of the changed menu item. If fuFlags is set to <u>MF_STRING</u> (the default), lpNewItem is a long pointer to a null-terminated string. If fuFlags is set to <u>MF_BITMAP</u> instead, lpNewItem contains a bitmap handle in its low-order word. If fuFlags is set to <u>MF_OWNERDRAW</u> , lpNewItem specifies an application-defined 32-bit value that the application can use to maintain additional data associated with the menu item. This 32-bit value is available to the application in the itemData member of the <u>MEASUREITEMSTRUCT</u> or <u>DRAWITEMSTRUCT</u> structure pointed to by the lParam parameter of the <u>WM_MEASUREITEM</u> or <u>WM_DRAWITEM</u> message. These messages are sent when the menu item is initially displayed or is changed. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

If the ModifyMenu function replaces a pop-up menu associated with the menu item, it destroys the old pop-up menu and frees the memory used by the pop-up menu.

Whenever a menu changes (whether or not it is in a window that is displayed), the application should call DrawMenuBar. To change the attributes of existing menu items, it is much faster to use the CheckMenuItem and EnableMenuItem functions.

Each of the following groups lists flags that should not be used together:

- MF\_BYCOMMAND and MF\_BYPOSITION
- MF\_DISABLED, MF\_ENABLED, and MF\_GRAYED
- MF\_BITMAP, MF\_STRING, MF\_OWNERDRAW, and MF\_SEPARATOR
- MF\_MENUBARBREAK and MF\_MENUBREAK
- MF\_CHECKED and MF\_UNCHECKED

The following list describes the flags that may be set in the fuFlags parameter:

| Value               | Meaning                                                                                                                                               |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>MF_BITMAP</u>    | Uses a bitmap as the menu item. The low-order word of the lpNewItem parameter contains the handle of the bitmap.                                      |
| <u>MF_BYCOMMAND</u> | Specifies that the idItem parameter gives the menu-item identifier. This is the default if neither <u>MF_BYCOMMAND</u> nor <u>MF_POSITION</u> is set. |

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MF_BYPOSITION   | Specifies that the <code>idItem</code> parameter gives the position of the menu item to be changed rather than the menu-item identifier.                                                                                                                                                                                                                                                                        |
| MF_CHECKED      | Places a check mark next to the menu item. If the application has supplied check-mark bitmaps (see <a href="#">SetMenuItemBitmaps</a> ), setting this flag displays the check-mark bitmap next to the menu item.                                                                                                                                                                                                |
| MF_DISABLED     | Disables the menu item so that it cannot be selected, but does not gray (dim) it.                                                                                                                                                                                                                                                                                                                               |
| MF_ENABLED      | Enables the menu item so that it can be selected and restores it from its grayed state.                                                                                                                                                                                                                                                                                                                         |
| MF_GRAYED       | Disables the menu item so that it cannot be selected and grays it.                                                                                                                                                                                                                                                                                                                                              |
| MF_MENUBARBREAK | Same as <a href="#">MF_MENUBREAK</a> except, for pop-up menus, separates the new column from the old column with a vertical line.                                                                                                                                                                                                                                                                               |
| MF_MENUBREAK    | Places the menu item on a new line for static menu-bar items. For pop-up menus, this flag places the item in a new column, with no dividing line between the columns.                                                                                                                                                                                                                                           |
| MF_OWNERDRAW    | Specifies that the menu item is an owner-drawn item. The window that owns the menu receives a <code>WM_MEASUREITEM</code> message when the menu is displayed for the first time to retrieve the height and width of the menu item. The <code>WM_DRAWITEM</code> message is then sent whenever the owner must update the visual appearance of the menu item. This option is not valid for a top-level menu item. |
| MF_POPUP        | Specifies that the item has a pop-up menu associated with it. The <code>idNewItem</code> parameter specifies a handle of a pop-up menu to be associated with the menu item. Use this flag for adding either a top-level pop-up menu or a hierarchical pop-up menu to a pop-up menu item.                                                                                                                        |
| MF_SEPARATOR    | Draws a horizontal dividing line. This line cannot be grayed, disabled, or highlighted. You can use this flag only in a pop-up menu. The <code>lpNewItem</code> and <code>idNewItem</code> parameters are ignored.                                                                                                                                                                                              |
| MF_STRING       | Specifies that the menu item is a character string; the <code>lpNewItem</code> parameter points to the string for the menu item.                                                                                                                                                                                                                                                                                |
| MF_UNCHECKED    | Does not select (place a check mark next to) the menu item. No check mark is the default condition if neither <a href="#">MF_CHECKED</a> nor <a href="#">MF_UNCHECKED</a> is set. If the application has supplied check-mark bitmaps (see the <a href="#">SetMenuItemBitmaps</a> function), setting this flag displays the "check mark off" bitmap next to the menu item.                                       |

See Also

[CheckMenuItem](#), [DrawMenuBar](#), [EnableMenuItem](#), [SetMenuItemBitmaps](#)

## MoveWindow (2.x)

```
BOOL MoveWindow(HWND hwnd, int nLeft, int nTop, int nWidth, int nHeight, BOOL fRepaint)
/* handle of window */
/* left coordinate */
/* top
coordinate
/
int nWidth;
* width
/
int nHeight;
* height
/
BOOL fRepaint;
* repaint flag
/
```

The MoveWindow function changes the position and dimensions of a window. For top-level windows, the position and dimensions are relative to the upper-left corner of the screen. For child windows, they are relative to the upper-left corner of the parent window's client area.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window to be changed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| nLeft     | Specifies the new position of the left side of the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| nTop      | Specifies the new position of the top of the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| nWidth    | Specifies the new width of the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| nHeight   | Specifies the new height of the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| fRepaint  | Specifies whether the window is to be repainted. If this parameter is TRUE, the window receives a WM_PAINT message as usual. If this parameter is FALSE, no repainting of any kind occurs. This applies to the client area, the non-client area (including the title and scroll bars), and any part of the parent window uncovered as a result of the moved window. When this parameter is FALSE, the application must explicitly invalidate or redraw any parts of the window and parent window that must be redrawn. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The MoveWindow function sends a WM\_GETMINMAXINFO message to the window being moved, giving it an opportunity to modify the default values for the largest and smallest possible windows. If the MoveWindow parameters exceed these values, they will be replaced by the minimum or maximum values specified in the WM\_GETMINMAXINFO message.

### Example

The following example changes the dimensions of a child window in response to a WM\_SIZE message. In this example, the child window would always fill the client area of the parent window.

```
case WM_SIZE:
    MoveWindow(hwndChild, 0, 0, LOWORD(lParam), HIWORD(lParam),
    TRUE);
    break;
```

### See Also

[ClientToScreen](#), [GetWindowRect](#), [ScreenToClient](#), [SetWindowPos](#), [WM\\_GETMINMAXINFO](#), [WM\\_SIZE](#)

### Windows 3.1 changes

For Windows version 3.0 applications, the `MoveWindow` function always paints the frame and erases the background of top-level windows, regardless of the setting of the `fRepaint` parameter.

## OffsetRect (2.x)

```
void OffsetRect(lprc, x, y)
RECT FAR* lprc;          /* address of structure with rectangle */
int x;                   /* horizontal offset */
/
int y;                   /
* vertical offset
/
```

The OffsetRect function moves the given rectangle by the specified offsets.

| Parameter | Description                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------|
| lprc      | Points to a <u>RECT</u> structure that contains the coordinates of the rectangle to be moved. |
| x         | Specifies the amount to move left or right. It must be negative to move left.                 |
| y         | Specifies the amount to move up or down. It must be negative to move up.                      |

### Returns

This function does not return a value.

### Comments

The coordinate values of a rectangle must not be greater than 32,767 or less than -32,768. The x and y parameters must be chosen carefully to prevent invalid rectangles.

### See Also

InflateRect, IntersectRect, UnionRect, RECT

## OpenClipboard (2.x)

BOOL OpenClipboard(hwnd)

HWND hwnd; /\* handle of window to associate ownership with \*/

The OpenClipboard function opens the clipboard. Other applications will not be able to modify the clipboard until the CloseClipboard function is called.

| Parameter | Description                                                     |
|-----------|-----------------------------------------------------------------|
| hwnd      | Identifies the window to be associated with the open clipboard. |

### Returns

The return value is nonzero if the function is successful. It is zero if another application or window has the clipboard opened.

### Comments

The window identified by the hwnd parameter will not become the owner of the clipboard until the EmptyClipboard function is called.

### See Also

CloseClipboard, EmptyClipboard

## OpenComm (2.x)

```
int OpenComm(lpszDevControl, cbInQueue, cbOutQueue)
LPCSTR lpszDevControl;      /* address of device-control information */
UINT cbInQueue;             /
/* size of receiving queue
/
UINT cbOutQueue;
/* size of transmission queue
/
```

The OpenComm function opens a communications device.

| Parameter      | Description                                                                                                          |
|----------------|----------------------------------------------------------------------------------------------------------------------|
| lpszDevControl | Points to a null-terminated string that specifies the device in the form COMn or LPTn, where n is the device number. |
| cbInQueue      | Specifies the size, in bytes, of the receiving queue. This parameter is ignored for LPT devices.                     |
| cbOutQueue     | Specifies the size, in bytes, of the transmission queue. This parameter is ignored for LPT devices.                  |

### Returns

The return value identifies the open device if the function is successful. Otherwise, it is less than zero.

### Errors

If the function fails, it may return one of the following error values:

| Value              | Meaning                                                      |
|--------------------|--------------------------------------------------------------|
| <u>IE_BADID</u>    | The device identifier is invalid or unsupported.             |
| <u>IE_BAUDRATE</u> | The device's baud rate is unsupported.                       |
| <u>IE_BYTESIZE</u> | The specified byte size is invalid.                          |
| <u>IE_DEFAULT</u>  | The default parameters are in error.                         |
| <u>IE_HARDWARE</u> | The hardware is not available (is locked by another device). |
| <u>IE_MEMORY</u>   | The function cannot allocate the queues.                     |
| <u>IE_NOPEN</u>    | The device is not open.                                      |
| <u>IE_OPEN</u>     | The device is already open.                                  |

If this function is called with both queue sizes set to zero, the return value is IE\_OPEN if the device is already open or IE\_MEMORY if the device is not open.

### Comments

Windows allows COM ports 1 through 9 and LPT ports 1 through 3. If the device driver does not support a communications port number, the OpenComm function will fail.

The communications device is initialized to a default configuration. The SetCommState function should be used to initialize the device to alternate values.

The receiving and transmission queues are used by interrupt-driven device drivers. LPT ports are not interrupt driven--for these ports, the cbInQueue and cbOutQueue parameters are ignored and the queue size is set to zero.

### Example

The following example uses the OpenComm function to open communications port 1:

```
idComDev = OpenComm("COM1", 1024, 128);
if (idComDev < 0) {
    ShowError(idComDev, "OpenComm");
    return 0;
}

err = BuildCommDCB("COM1:9600,n,8,1", &dcb);
if (err < 0) {
    ShowError(err, "BuildCommDCB");
    return 0;
}
```

```
err = SetCommState(&dcb);  
if (err < 0) {  
    ShowError(err, "SetCommState");  
    return 0;  
}
```

See Also

CloseComm, SetCommState



IE\_BADID (-1)

The device identifier is invalid or unsupported.

IE\_BADID (-1)

IE\_BAUDRATE (-12)

The device's baud rate is unsupported.

IE\_BAUDRATE (-12)

IE\_BYTESIZE (-11)

The specified byte size is invalid.

IE\_BYTESIZE (-11)

IE\_DEFAULT (-5)

The default parameters are in error.

IE\_DEFAULT (-5)



IE\_HARDWARE (-10)

The hardware is not available (is locked by another device).

IE\_HARDWARE (-10)

IE\_MEMORY (-4)

The function cannot allocate the queues.

IE\_MEMORY (-4)

IE\_NOPEN (-3)

The device is not open.

IE\_NOPEN (-3)

IE\_OPEN (-2)

The device is already open.

IE\_OPEN (-2)



## OpenDriver (3.1)

HDRVR OpenDriver(lpDriverName, lpSectionName, lParam)

LPCSTR lpDriverName; /\* address of driver name \*/

LPCSTR lpSectionName;

\* address of .INI file section name

/

LPARAM lParam;

\* address of driver-specific information

/

The `OpenDriver` function performs necessary initialization operations such as setting members in installable-driver structures to their default values.

| Parameter                  | Description                                                                                     |
|----------------------------|-------------------------------------------------------------------------------------------------|
| <code>lpDriverName</code>  | Points to a null-terminated string that specifies the name of an installable driver.            |
| <code>lpSectionName</code> | Points to a null-terminated string that specifies the name of a section in the SYSTEM.INI file. |
| <code>lParam</code>        | Specifies driver-specific information.                                                          |

### Returns

The return value is a handle of the installable driver, if the function is successful. Otherwise it is `NULL`.

### Comments

The string to which `lpDriverName` points must be identical to the name of the installable driver as it appears in the SYSTEM.INI file.

If the name of the installable driver appears in the [driver] section of the SYSTEM.INI file, the string pointed to by `lpSectionName` should be `NULL`. Otherwise this string should specify the name of the section in SYSTEM.INI that contains the driver name.

When an application opens a driver for the first time, Windows calls the `DriverProc` function with the `DRV_LOAD`, `DRV_ENABLE`, and `DRV_OPEN` messages. When subsequent instances of the driver are opened, only `DRV_OPEN` is sent.

The value specified in the `lParam` parameter is passed to the `lParam2` parameter of the `DriverProc` function.

### See Also

[CloseDriver](#), [DriverProc](#)

## OpenIcon (2.x)

```
BOOL OpenIcon(hwnd)  
HWND hwnd; /* handle of window */
```

The `OpenIcon` function activates and displays a minimized window. Windows restores the window to its original size and position.

| Parameter         | Description            |
|-------------------|------------------------|
| <code>hwnd</code> | Identifies the window. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

Using `OpenIcon` is the same as specifying the `SW_SHOWNORMAL` flag in a call to the [ShowWindow](#) function.

### See Also

[CloseWindow](#), [IsIconic](#), [ShowWindow](#)

## PeekMessage (2.x)

```
BOOL PeekMessage(lpmsg, hwnd, uFilterFirst, uFilterLast, fuRemove)
MSG FAR* lpmsg;      /* address of structure for message */
HWND hwnd;          /* handle of
filter window
/
UINT uFilterFirst;
* first message
/
UINT uFilterLast;
* last message
/
UINT fuRemove;
* removal flags
/
```

The PeekMessage function checks the application's message queue for a message and places the message (if any) in the specified MSG structure.

| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                    |                                                                          |                   |                                                                                       |                  |                                                                      |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------------------|--------------------------------------------------------------------------|-------------------|---------------------------------------------------------------------------------------|------------------|----------------------------------------------------------------------|
| lpmsg              | Points to an <u>MSG</u> structure that will receive message information from the application's message queue.                                                                                                                                                                                                                                                                                                                                             |       |         |                    |                                                                          |                   |                                                                                       |                  |                                                                      |
| hwnd               | Identifies the window whose messages are to be examined.                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                    |                                                                          |                   |                                                                                       |                  |                                                                      |
| uFilterFirst       | Specifies the value of the first message in the range of messages to be examined.                                                                                                                                                                                                                                                                                                                                                                         |       |         |                    |                                                                          |                   |                                                                                       |                  |                                                                      |
| uFilterLast        | Specifies the value of the last message in the range of messages to be examined.                                                                                                                                                                                                                                                                                                                                                                          |       |         |                    |                                                                          |                   |                                                                                       |                  |                                                                      |
| fuRemove           | Specifies how messages are handled. This parameter can be a combination of the following values ( <u>PM_NOYIELD</u> can be combined with either <u>PM_NOREMOVE</u> or <u>PM_REMOVE</u> ):                                                                                                                                                                                                                                                                 |       |         |                    |                                                                          |                   |                                                                                       |                  |                                                                      |
|                    | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>PM_NOREMOVE</u></td><td>Messages are not removed from the queue after processing by PeekMessage.</td></tr><tr><td><u>PM_NOYIELD</u></td><td>Prevents the current task from halting and yielding system resources to another task.</td></tr><tr><td><u>PM_REMOVE</u></td><td>Messages are removed from the queue after processing by PeekMessage.</td></tr></tbody></table> | Value | Meaning | <u>PM_NOREMOVE</u> | Messages are not removed from the queue after processing by PeekMessage. | <u>PM_NOYIELD</u> | Prevents the current task from halting and yielding system resources to another task. | <u>PM_REMOVE</u> | Messages are removed from the queue after processing by PeekMessage. |
| Value              | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                    |                                                                          |                   |                                                                                       |                  |                                                                      |
| <u>PM_NOREMOVE</u> | Messages are not removed from the queue after processing by PeekMessage.                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                    |                                                                          |                   |                                                                                       |                  |                                                                      |
| <u>PM_NOYIELD</u>  | Prevents the current task from halting and yielding system resources to another task.                                                                                                                                                                                                                                                                                                                                                                     |       |         |                    |                                                                          |                   |                                                                                       |                  |                                                                      |
| <u>PM_REMOVE</u>   | Messages are removed from the queue after processing by PeekMessage.                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                    |                                                                          |                   |                                                                                       |                  |                                                                      |

### Returns

The return value is nonzero if a message is available. Otherwise, it is zero.

### Comments

Unlike the GetMessage function, the PeekMessage function does not wait for a message to be placed in the queue before returning. PeekMessage yields control to other tasks, unless the PM\_NOYIELD flag is set. However, if there is a WM\_TIMER message pending, PeekMessage will yield regardless of the PM\_NOYIELD flag.

PeekMessage retrieves only messages associated with the window identified by the hwnd parameter, or any of its children as specified by the IsChild function, and within the range of message values given by the uFilterFirst and uFilterLast parameters. If hwnd is NULL, PeekMessage retrieves messages for any window that belongs to the application making the call. (PeekMessage does not retrieve messages for windows that belong to other applications.) If uFilterFirst and uFilterLast are both zero, PeekMessage returns all available messages (no range filtering is performed).

The WM\_KEYFIRST and WM\_KEYLAST flags can be used as filter values to retrieve all key messages; the WM\_MOUSEFIRST and WM\_MOUSELAST flags can be used to retrieve all mouse messages.

PeekMessage does not remove WM\_PAINT messages from the queue. The messages remain in the queue until processed. The GetMessage, PeekMessage, and WaitMessage functions yield control to other applications. These calls provide the only way to let other applications run. If your application does not call any of these functions for long periods of time, other applications cannot run.

As long as an application is in a PeekMessage loop, Windows cannot become idle. Therefore, an application should not remain in a PeekMessage loop after the application's background processing has completed.

When an application uses the `PeekMessage` function without removing the message and then calls the `WaitMessage` function, `WaitMessage` does not return until the message is received. Applications that use the `PeekMessage` function should remove any retrieved messages from the queue before calling `WaitMessage`.

#### Example

The following example checks the message queue for keystrokes that have special meaning to the application. Note that the `CheckSpecialKeys` function is application-defined.

```
MSG msg;
BOOL fRetVal = TRUE;

while (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE)) {

    if (msg.message == WM_QUIT)
        fRetVal = FALSE;

    if (CheckSpecialKeys(&msg)) /* application defined */
        continue;

    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
return fRetVal;
```

See Also

[GetMessage](#), [IsChild](#), [PostAppMessage](#), [SetMessageQueue](#), [WaitMessage](#)

## Corrections

Previous documentation incorrectly stated that a -1 could be used for the hwnd parameter. This parameter can only be NULL or a valid window handle.

PM\_NOREMOVE 0x0000

Messages are not removed from the queue after processing by PeekMessage.

PM\_NOREMOVE 0x0000

PM\_NOYIELD 0x0002

Prevents the current task from halting and yielding system resources to another task.



PM\_NOYIELD 0x0002

PM\_REMOVE 0x0001

Messages are removed from the queue after processing by PeekMessage.

PM\_REMOVE 0x0001

## PostAppMessage (2.x)

```
BOOL PostAppMessage(htask, uMsg, wParam, lParam)
HTASK htask;          /* handle of task to receive message */
UINT uMsg;           /* message to post */
post
/
WPARAM wParam;
* first message parameter
/
LPARAM lParam;
* second message parameter
/
```

The PostAppMessage function posts (places) a message in the message queue of the given application (task) and then returns without waiting for the application to process the message. The application to which the message is posted retrieves the message by calling the GetMessage or PeekMessage function. The hwnd member of the returned MSG structure is NULL.

| Parameter | Description                                                                                                 |
|-----------|-------------------------------------------------------------------------------------------------------------|
| htask     | Identifies the task to which the message is posted. The <u>GetCurrentTask</u> function returns this handle. |
| uMsg      | Specifies the type of message to be posted.                                                                 |
| wParam    | Specifies 16 bits of additional message-dependent information.                                              |
| lParam    | Specifies 32 bits of additional message-dependent information.                                              |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### See Also

GetCurrentTask, GetMessage, PeekMessage, PostMessage, MSG

■ **PostMessage (2.x)**

```
BOOL PostMessage(hwnd, uMsg, wParam, lParam)
HWND hwnd;          /* handle of the destination window */
UINT uMsg;          /* message to post */
/* first message parameter */
/* second message parameter */
```

The PostMessage function posts (places) a message in a window's message queue and then returns without waiting for the corresponding window to process the message. Messages in a message queue are retrieved by calls to the GetMessage or PeekMessage function.

| Parameter | Description                                                                                                                                                                                                         |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window to which the message will be posted. If this parameter is <code>HWND_BROADCAST</code> , the message will be posted to all top-level windows, including disabled or invisible unowned windows. |
| uMsg      | Specifies the message to be posted.                                                                                                                                                                                 |
| wParam    | Specifies 16 bits of additional message-dependent information.                                                                                                                                                      |
| lParam    | Specifies 32 bits of additional message-dependent information.                                                                                                                                                      |

**Returns**

The return value is nonzero if the function is successful. Otherwise, it is zero.

**Comments**

An application should never use the PostMessage function to post a message to a control.

If the message is being posted to another application, and the wParam or lParam parameters are used to pass a handle or pointer to a global memory object, the memory should be allocated by the GlobalAlloc function, using the GMEM\_SHARE flag.

The PostMessage function fails if the message queue for the receiving application is full. This is especially likely if an application posts several messages without allowing the receiving task to run. The GetMessage, PeekMessage, and WaitMessage functions yield control to other applications.

**See Also**

GetMessage, PeekMessage, PostAppMessage, SendDlgItemMessage, SendMessage

### Windows 3.1 changes

In Windows 3.0, specifying a window handle of -1 would not send the message to disabled or invisible unowned windows. In Windows 3.1, if a window handle of `HWND_BROADCAST` (-1) is specified, the message is sent to disabled or invisible unowned windows.

## PostQuitMessage (2.x)

```
void PostQuitMessage(nExitCode)
int nExitCode;    /* exit code */
```

The PostQuitMessage function posts a message to Windows indicating that an application is requesting to terminate execution (quit). This function is typically used in response to a WM\_DESTROY message.

| Parameter | Description                                                                                                |
|-----------|------------------------------------------------------------------------------------------------------------|
| nExitCode | Specifies an application-defined exit code. It must be the wParam parameter of the <u>WM_QUIT</u> message. |

### Returns

This function does not return a value.

### Comments

The PostQuitMessage function posts a WM\_QUIT message to the application and returns immediately; the function simply indicates to the system that the application will request to quit some time in the future.

When the application receives the WM\_QUIT message, it should exit the message loop in the main function and return control to Windows.

### See Also

GetMessage, WM\_DESTROY, WM\_QUIT

## PtInRect (2.x)

```
BOOL PtInRect(lprc, pt)
const RECT FAR* lprc;          /* address of structure with rectangle */
POINT pt;                      /* structure with
point                            */
/
```

The PtInRect function determines whether the specified point lies within a given rectangle. A point is within a rectangle if it lies on the left or top side or is within all four sides. A point on the right or bottom side is considered outside the rectangle.

| Parameter | Description                                                              |
|-----------|--------------------------------------------------------------------------|
| lprc      | Points to a <u>RECT</u> structure that contains the specified rectangle. |
| pt        | Specifies a <u>POINT</u> structure that contains the specified point.    |

### Returns

The return value is nonzero if the point lies within the given rectangle. Otherwise, it is zero.

### See Also

EqualRect, IsRectEmpty, POINT, RECT



### QuerySendMessage (3.1)

```
BOOL QuerySendMessage(hreserved1, hreserved2, hreserved3, lpMessage)
HANDLE hreserved1;
HANDLE hreserved2;
HANDLE hreserved3;
LPMSG lpMessage; /* address of
structure for message
/
```

The QuerySendMessage function determines whether a message sent by SendMessage originated from within the current task. If the message is an intertask message, QuerySendMessage puts it into the specified MSG structure.

| Parameter  | Description                                                                |
|------------|----------------------------------------------------------------------------|
| hreserved1 | Reserved; must be NULL.                                                    |
| hreserved2 | Reserved; must be NULL.                                                    |
| hreserved3 | Reserved; must be NULL.                                                    |
| lpMessage  | Specifies the <u>MSG</u> structure in which to place an intertask message. |

#### Returns

The return value is zero if the message originated within the current task. Otherwise, it is nonzero.

#### Comments

If the Windows debugger is entering soft mode, the application being debugged should reply to intertask messages by using the ReplyMessage function.

The NULL parameters are reserved for future use.

#### See Also

SendMessage, ReplyMessage, MSG

## ReadComm (2.x)

```
int ReadComm(idComDev, lpvBuf, cbRead)
int idComDev;          /* identifier of device to read from */
void FAR* lpvBuf;     /* address of
buffer for read bytes
/
int cbRead;
* number of bytes to read
/
```

The ReadComm function reads up to a specified number of bytes from the given communications device.

| Parameter | Description                                                                                           |
|-----------|-------------------------------------------------------------------------------------------------------|
| idComDev  | Specifies the communications device to be read from. The <u>OpenComm</u> function returns this value. |
| lpvBuf    | Points to the buffer for the read bytes.                                                              |
| cbRead    | Specifies the number of bytes to be read.                                                             |

### Returns

The return value is the number of bytes read, if the function is successful. Otherwise, it is less than zero and its absolute value is the number of bytes read.

For parallel I/O ports, the return value is always zero.

### Comments

When an error occurs, the cause of the error can be determined by using the GetCommError function to retrieve the error value and status. Since errors can occur when no bytes are present, if the return value is zero, the GetCommError function should be used to ensure that no error occurred.

The return value is less than the number specified by the cbRead parameter only if the number of bytes in the receiving queue is less than that specified by cbRead. If the return value is equal to cbRead, additional bytes may be queued for the device. If the return value is zero, no bytes are present.

### See Also

GetCommError, OpenComm

## RealizePalette (3.0)

```
UINT RealizePalette(hdc)
HDC hdc; /* handle of device context */
```

The RealizePalette function maps palette entries from the current logical palette to the system palette.

| Parameter | Description                                                 |
|-----------|-------------------------------------------------------------|
| hdc       | Identifies the device context containing a logical palette. |

### Returns

The return value indicates how many entries in the logical palette were mapped to different entries in the system palette. This represents the number of entries that this function remapped to accommodate changes in the system palette since the logical palette was last realized.

### Comments

A logical color palette acts as a buffer between color-intensive applications and the system, allowing an application to use as many colors as necessary without interfering with either its own displayed color or with colors displayed by other windows. When a window has the input focus and calls the RealizePalette function, Windows ensures that the window will display all the requested colors (up to the maximum number simultaneously available on the screen) and Windows displays additional colors by matching them to available colors. In addition, Windows matches the colors requested by inactive windows that call RealizePalette as closely as possible to the available colors. This significantly reduces undesirable changes in the colors displayed in inactive windows.

### Example

The following example uses the [SelectPalette](#) function to select a palette into a device context and then calls the RealizePalette function to map the colors to the system palette:

```
HPALETTE hpal, hPalPrevious;

hdc = GetDC(hwnd);

hPalPrevious = SelectPalette(hdc, hpal, FALSE);
if (RealizePalette(hdc) == NULL)
    MessageBox(hwnd, "Can't realize palette", "Error", MB_OK);

ReleaseDC(hwnd, hdc);
```

### See Also

[SelectPalette](#), [WM\\_PALETTECHANGED](#)

### RedrawWindow (3.1)

```

BOOL RedrawWindow(hwnd, lprcUpdate, hrgnUpdate, fuRedraw)
HWND hwnd; /* handle of window */
const RECT FAR* lprcUpdate; /* address of structure with update rect. */
HRGN hrgnUpdate; /* handle of update region */
UINT fuRedraw; /* redraw flags */

```

The RedrawWindow function updates the specified rectangle or region in the given window's client area.

| Parameter  | Description                                                                                                                                                                 |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd       | Identifies the window to be redrawn. If this parameter is NULL, the desktop window is updated.                                                                              |
| lprcUpdate | Points to a <u>RECT</u> structure containing the coordinates of the update rectangle. This parameter is ignored if the hrgnUpdate parameter contains a valid region handle. |
| hrgnUpdate | Identifies the update region. If both the hrgnUpdate and lprcUpdate parameters are NULL, the entire client area is added to the update region.                              |
| fuRedraw   | Specifies one or more redraw flags. This parameter can be a combination of flags: The following flags are used to invalidate the window:                                    |

| Value                    | Meaning                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>RDW_ERASE</u>         | Causes the window to receive a <u>WM_ERASEBKGND</u> message when the window is repainted. The <u>RDW_INVALIDATE</u> flag must also be specified; otherwise, <u>RDW_ERASE</u> has no effect.                                                                                                                                                                                                                    |
| <u>RDW_FRAME</u>         | Causes any part of the non-client area of the window that intersects the update region to receive a <u>WM_NCPAINT</u> message. The <u>RDW_INVALIDATE</u> flag must also be specified; otherwise, <u>RDW_FRAME</u> has no effect. The <u>WM_NCPAINT</u> message is typically not sent during the execution of the RedrawWindow function unless either <u>RDW_UPDATENOW</u> or <u>RDW_ERASENOW</u> is specified. |
| <u>RDW_INTERNALPAINT</u> | Causes a <u>WM_PAINT</u> message to be posted to the window regardless of whether the window contains an invalid region.                                                                                                                                                                                                                                                                                       |
| <u>RDW_INVALIDATE</u>    | Invalidate lprcUpdate or hrgnUpdate (only one may be non-NULL). If both are NULL, the entire window is invalidated.                                                                                                                                                                                                                                                                                            |

The following flags are used to validate the window:

| Value                      | Meaning                                                                                                                                                                                                                                                 |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>RDW_NOERASE</u>         | Suppresses any pending <u>WM_ERASEBKGND</u> messages.                                                                                                                                                                                                   |
| <u>RDW_NOFRAME</u>         | Suppresses any pending <u>WM_NCPAINT</u> messages. This flag must be used with <u>RDW_VALIDATE</u> and is typically used with <u>RDW_NOCHILDREN</u> . This option should be used with care, as it could cause parts of a window from painting properly. |
| <u>RDW_NOINTERNALPAINT</u> | Suppresses any pending internal <u>WM_PAINT</u> messages. This flag does not affect <u>WM_PAINT</u> messages resulting from invalid areas.                                                                                                              |

RDW\_VALIDATE Validates lprcUpdate or hrgnUpdate (only one may be non-NULL). If both are NULL, the entire window is validated. This flag does not affect internal WM\_PAINT messages.

The following flags control when repainting occurs. No painting is performed by the RedrawWindow function unless one of these bits is specified.

| Value                | Meaning                                                                                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>RDW_ERASENOW</u>  | Causes the affected windows (as specified by the <u>RDW_ALLCHILDREN</u> and <u>RDW_NOCHILDREN</u> flags) to receive <u>WM_NCPAINT</u> and <u>WM_ERASEBKGD</u> messages, if necessary, before the function returns. <u>WM_PAINT</u> messages are deferred. |
| <u>RDW_UPDATENOW</u> | Causes the affected windows (as specified by the <u>RDW_ALLCHILDREN</u> and <u>RDW_NOCHILDREN</u> flags) to receive <u>WM_NCPAINT</u> , <u>WM_ERASEBKGD</u> , and <u>WM_PAINT</u> messages, if necessary, before the function returns.                    |

By default, the windows affected by the RedrawWindow function depend on whether the specified window has the WS\_CLIPCHILDREN style. The child windows of WS\_CLIPCHILDREN windows are not affected; however, non-WS\_CLIPCHILDREN windows are recursively validated or invalidated until a WS\_CLIPCHILDREN window is encountered. The following flags control which windows are affected by the RedrawWindow function:

| Value                  | Meaning                                                        |
|------------------------|----------------------------------------------------------------|
| <u>RDW_ALLCHILDREN</u> | Includes child windows, if any, in the repainting operation.   |
| <u>RDW_NOCHILDREN</u>  | Excludes child windows, if any, from the repainting operation. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

When the RedrawWindow function is used to invalidate part of the desktop window, the desktop window does not receive a WM\_PAINT message. To repaint the desktop, an application should use the RDW\_ERASE flag to generate a WM\_ERASEBKGD message.

#### See Also

GetUpdateRect, GetUpdateRgn, InvalidateRect, InvalidateRgn, UpdateWindow, WM\_ERASEBKGD, WM\_PAINT

RDW\_ERASE 0x0004

Causes the window to receive a WM\_ERASEBKGND message when the window is repainted. The RDW\_INVALIDATE flag must also be specified; otherwise, RDW\_ERASE has no effect.

RDW\_ERASE 0x0004

RDW\_FRAME 0x0400

Causes any part of the non-client area of the window that intersects the update region to receive a WM\_NCPAINT message. The RDW\_INVALIDATE flag must also be specified; otherwise, RDW\_FRAME has no effect. The WM\_NCPAINT message is typically not sent during the execution of the RedrawWindow function unless either RDW\_UPDATENOW or RDW\_ERASENOW is specified.



RDW\_FRAME 0x0400

RDW\_INTERNALPAINT 0x0002

Causes a WM\_PAINT message to be posted to the window regardless of whether the window contains an invalid region.

RDW\_INTERNALPAINT 0x0002

RDW\_INVALIDATE 0x0001

Invalidate lprcUpdate or hrgnUpdate (only one may be non-NULL). If both are NULL, the entire window is invalidated.

RDW\_INVALIDATE 0x0001

RDW\_NOERASE 0x0020

Suppresses any pending WM\_ERASEBKGD messages.

RDW\_NOERASE 0x0020

RDW\_NOFRAME 0x0800

Suppresses any pending WM\_NCPAINT messages. This flag must be used with RDW\_VALIDATE and is typically used with RDW\_NOCHILDREN. This option should be used with care, as it could cause parts of a window from painting properly.



RDW\_NOFRAME 0x0800

RDW\_NOINTERNALPAINT 0x0010

Suppresses any pending internal WM\_PAINT messages. This flag does not affect WM\_PAINT messages resulting from invalid areas.

RDW\_NOINTERNALPAINT 0x0010

RDW\_VALIDATE 0x0008

Validates lprcUpdate or hrgnUpdate (only one may be non-NULL). If both are NULL, the entire window is validated. This flag does not affect internal WM\_PAINT messages.

RDW\_VALIDATE 0x0008

RDW\_ERASENOW 0x0200

Causes the affected windows (as specified by the RDW\_ALLCHILDREN and RDW\_NOCHILDREN flags) to receive WM\_NCPAINT and WM\_ERASEBKGD messages, if necessary, before the function returns. WM\_PAINT messages are deferred.

RDW\_ERASENOW 0x0200

RDW\_UPDATENOW 0x0100

Causes the affected windows (as specified by the RDW\_ALLCHILDREN and RDW\_NOCHILDREN flags) to receive WM\_NCPAINT, WM\_ERASEBKGD, and WM\_PAINT messages, if necessary, before the function returns.



RDW\_UPDATENOW 0x0100

RDW\_ALLCHILDREN 0x0080

Includes child windows, if any, in the repainting operation.

RDW\_ALLCHILDREN 0x0080

RDW\_NOCHILDREN 0x0040

Excludes child windows, if any, from the repainting operation.

RDW\_NOCHILDREN 0x0040

## RegisterClass (2.x)

```
ATOM RegisterClass(lpwc)
const WNDCLASS FAR* lpwc;          /* address of structure with class data */
```

The RegisterClass function registers a window class for subsequent use in calls to the [CreateWindow](#) or [CreateWindowEx](#) function.

| Parameter | Description                                                                                                                                             |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpwc      | Points to a <a href="#">WNDCLASS</a> structure. The structure must be filled with the appropriate class attributes before being passed to the function. |

### Returns

The return value is an atom that uniquely identifies the class being registered. For Windows versions 3.0 and earlier, the return value is nonzero if the function is successful or zero if an error occurs.

### Comments

An application cannot register a global class if either a global class or a task-specific class already exists with the given name.

An application can register a task-specific class with the same name as a global class. The task-specific class overrides the global class for the current task only. A task cannot register two local classes with the same name. However, two different tasks can register task-specific classes using the same name.

### Example

The following example registers a window class, then creates a window of that class:

```
WNDCLASS wc;
HINSTANCE hinst;
char szMyClass[] = "MyClass";
HWND hwndMyWindow;

/* Register the window class. */

wc.style      = 0;
wc.lpfnWndProc = MyWndProc;
wc.cbClsExtra  = 0;
wc.cbWndExtra  = 0;
wc.hInstance = hinst;
wc.hIcon      = LoadIcon(hinst, "MyIcon");
wc.hCursor    = LoadCursor(NULL, IDC_ARROW);
wc.hbrBackground = (HBRUSH) (COLOR_WINDOW + 1);
wc.lpszMenuName = NULL;
wc.lpszClassName = szMyClass;

if (!RegisterClass(&wc))
    return FALSE;

/* Create the window. */

hwndMyWindow = CreateWindow(szMyClass, "MyApp",
    WS_OVERLAPPED | WS_SYSMENU, CW_USEDEFAULT, 0,
    CW_USEDEFAULT, 0, NULL, NULL,
    hinst, NULL );
```

### See Also

[CreateWindow](#), [CreateWindowEx](#), [GetClassInfo](#), [GetClassName](#), [UnregisterClass](#), [WindowProc](#), [WNDCLASS](#)

### Windows 3.1 changes

The RegisterClass function returns an atom that uniquely identifies the class being registered. For Windows version 3.0 and earlier, the return value is nonzero if the function is successful or zero if an error occurs.

## RegisterClipboardFormat (2.x)

```
UINT RegisterClipboardFormat(lpszFormatName)
LPCSTR lpszFormatName;          /* address of name string */
```

The RegisterClipboardFormat function registers a new clipboard format. The registered format can be used in subsequent clipboard functions as a valid format in which to render data, and it will appear in the clipboard's list of formats.

| Parameter      | Description                                                   |
|----------------|---------------------------------------------------------------|
| lpszFormatName | Points to a null-terminated string that names the new format. |

### Returns

The return value indicates the newly registered format. If the identical format name has been registered before, even by a different application, the format's reference count is incremented (increased by one) and the same value is returned as when the format was originally registered. The return value is zero if the format cannot be registered.

### Comments

The format value returned by the RegisterClipboardFormat function is within the range 0xC000 through 0xFFFF.

### See Also

[CountClipboardFormats](#), [EnumClipboardFormats](#), [GetClipboardFormatName](#),  
[GetPriorityClipboardFormat](#), [IsClipboardFormatAvailable](#)



## RegisterWindowMessage (2.x)

```
UINT RegisterWindowMessage(lpsz)
LPCSTR lpsz;          /* address of message string */
```

The RegisterWindowMessage function defines a new window message that is guaranteed to be unique throughout the system. The returned message value can be used when calling the [SendMessage](#) or [PostMessage](#) function.

| Parameter | Description                                                                     |
|-----------|---------------------------------------------------------------------------------|
| lpsz      | Points to a null-terminated string that specifies the message to be registered. |

### Returns

The return value is an unsigned short integer in the range 0xC000 through 0xFFFF if the message is successfully registered. Otherwise, the return value is 0.

### Comments

RegisterWindowMessage is typically used to register messages for communicating between two cooperating applications.

If two different applications register the same message string, the applications return the same message value. The message remains registered until the Windows session ends.

Use the RegisterWindowMessage function only when more than one application must process the same message. For sending private messages within a window class, an application can use any integer in the range [WM\\_USER](#) through 0x7FFF. (Messages in this range are private to a window class, not to an application. For example, such predefined control classes as [BUTTON](#), [EDIT](#), [LISTBOX](#), and [COMBOBOX](#) may use values in this range.)

### See Also

[PostAppMessage](#), [PostMessage](#), [SendMessage](#)

## ReleaseCapture (2.x)

void ReleaseCapture(void)

The ReleaseCapture function releases the mouse capture and restores normal input processing. A window with the mouse capture receives all mouse input regardless of the position of the cursor.

### Returns

This function does not return a value.

### Comments

An application calls this function after calling the SetCapture function.

### See Also

SetCapture

## ReleaseDC (2.x)

```
int ReleaseDC(hwnd, hdc)
HWND hwnd;    /* handle of window with device context */
HDC hdc;      /* handle of device context */
context
/
```

The ReleaseDC function releases the given device context, freeing it for use by other applications.

| Parameter | Description                                                   |
|-----------|---------------------------------------------------------------|
| hwnd      | Identifies the window whose device context is to be released. |
| hdc       | Identifies the device context to be released.                 |

### Returns

The return value is 1 if the function is successful. Otherwise, it is 0.

### Comments

The effect of ReleaseDC depends on the type of device context. It frees only common and window device contexts. It has no effect on class or private device contexts.

The application must call the ReleaseDC function for each call to the GetWindowDC function and for each call to the GetDC function that retrieves a common device context.

### See Also

BeginPaint, EndPaint, GetDC, GetWindowDC

## RemoveMenu (3.0)

```
BOOL RemoveMenu(hmenu, idItem, fuFlags)
HMENU hmenu;      /* handle of menu      */
UINT idItem;      /* menu item to delete */
UINT fuFlags;
/* menu flags
/
```

The RemoveMenu function deletes a menu item with an associated pop-up menu from a menu but does not destroy the handle of the pop-up menu, allowing the menu to be reused. Before calling this function, an application should call the GetSubMenu function to retrieve the pop-up menu handle.

| Parameter | Description                                                                                                 |
|-----------|-------------------------------------------------------------------------------------------------------------|
| hmenu     | Identifies the menu to be changed.                                                                          |
| idItem    | Specifies the menu item to be removed, as determined by the fuFlags parameter.                              |
| fuFlags   | Specifies how the idItem parameter is to be interpreted. This parameter can be one of the following values: |

| Value         | Meaning                                                                  |
|---------------|--------------------------------------------------------------------------|
| MF_BYCOMMAND  | The idItem parameter specifies the menu-item identifier.                 |
| MF_BYPOSITION | The idItem parameter specifies the zero-based position of the menu item. |

### Returns

The return value is nonzero if the function is successful. Otherwise it is zero.

### Comments

Whenever a menu changes (whether or not it is in a window that is displayed), the application should call the DrawMenuBar function.

### See Also

AppendMenu, CreateMenu, DeleteMenu, DrawMenuBar, GetSubMenu, InsertMenu

## RemoveProp (2.x)

```
HANDLE RemoveProp(hwnd, lpsz)
HWND hwnd;          /* handle of window */
LPCSTR lpsz;        /* atom or address of string */
```

The RemoveProp function removes an entry from the property list of the given window. The RemoveProp function returns a data handle so that the application can free the data associated with the handle.

| Parameter | Description                                                                                                                                                                                                                                                                                                |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window whose property list is to be changed.                                                                                                                                                                                                                                                |
| lpsz      | Points to a null-terminated string or an atom that identifies a string. If an atom is given, it must be a global atom created by a previous call to the <u>GlobalAddAtom</u> function. The atom, a 16-bit value, must be placed in the low-order word of this parameter; the high-order word must be zero. |

### Returns

The return value is the handle of the given string if the function is successful. Otherwise, it is NULL--for example, if the string cannot be found in the given property list.

### Comments

An application can remove only those properties it has added. It should not remove properties added by other applications or by Windows itself.

An application must free the data handles associated with entries removed from a property list. The application should remove only those properties it added to the property list.

### See Also

GetProp, GlobalAddAtom

## ReplyMessage (2.x)

```
void ReplyMessage(IResult)
LRESULT IResult;      /* message-dependent reply */
```

The ReplyMessage function is used to reply to a message sent through the SendMessage function without returning control to the function that called SendMessage.

| Parameter | Description                                                                                     |
|-----------|-------------------------------------------------------------------------------------------------|
| IResult   | Specifies the result of the message processing. The possible values depend on the message sent. |

### Returns

This function does not return a value.

### Comments

By calling this function, the window procedure that receives the message allows the task that called SendMessage to continue to run as though the task that received the message had returned control. The task that calls ReplyMessage also continues to run.

Usually, a task that calls SendMessage to send a message to another task will not continue running until the window procedure that Windows calls to receive the message returns. However, if a task that is called to receive a message must perform some type of operation that might yield control (such as calling the MessageBox or DialogBox function), Windows could be deadlocked, as when the sending task must run and process messages but cannot because it is waiting for SendMessage to return. An application can avoid this problem if the task receiving the message calls ReplyMessage before performing any operation that could cause the task to yield.

The ReplyMessage function has no effect if the message was not sent through the SendMessage function or if the message was sent by the same task.

### See Also

DialogBox, MessageBox, SendMessage

## ScreenToClient (2.x)

```
void ScreenToClient(hwnd, lppt)
HWND hwnd;          /* window handle for source coordinates */
POINT FAR* lppt;    /*
address of structure with coordinates
/
```

The ScreenToClient function converts the screen coordinates of a given point on the screen to client coordinates.

| Parameter | Description                                                                              |
|-----------|------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window whose client area is to be used for the conversion.                |
| lppt      | Points to a <u>POINT</u> structure that contains the screen coordinates to be converted. |

Returns  
This function does not return a value.

Comments  
The ScreenToClient function replaces the screen coordinates in the POINT structure with client coordinates. The new coordinates are relative to the upper-left corner of the given window's client area.

Example  
The following example uses the GetWindowRect function to retrieve the screen coordinates for a specified window, calls the ScreenToClient function to convert the upper-left and lower-right corners of the window rectangle to client coordinates, and then reports the results in a message box:

```
RECT rc; /* window's screen coordinates */
POINT ptUpperLeft; /* client coordinate of upper left */
POINT ptLowerRight; /* client coordinate of lower right */
char szText[128]; /* char buffer for wsprintf */

GetWindowRect(hwnd, &rc);

ptUpperLeft.x = rc.left;
ptUpperLeft.y = rc.top;
ptLowerRight.x = rc.right;
ptLowerRight.y = rc.bottom;

ScreenToClient(hwnd, &ptUpperLeft );
ScreenToClient(hwnd, &ptLowerRight);

wsprintf(szText,
         "S: (%d,%d)-(%d,%d) --> C: (%d,%d)-(%d,%d)",
         rc.left, rc.top, rc.right, rc.bottom,
         ptUpperLeft.x, ptUpperLeft.y, ptLowerRight.x, ptLowerRight.y);

MessageBox(hwnd, szText, "ScreenToClient", MB_OK);
```

See Also  
ClientToScreen, MapWindowPoints, POINT

## ScrollDC (2.x)

```
BOOL ScrollDC(hdc, dx, dy, lprcScroll, lprcClip, hrgnUpdate, lprcUpdate)
HDC hdc; /* handle of device context */
int dx; /* horizontal scroll units */
int dy; /* vertical scroll units */
const RECT FAR* lprcScroll; /* address of scrolling rectangle */
const RECT FAR* lprcClip; /* address of clipping rectangle */
HRGN hrgnUpdate; /* handle of scrolling region */
RECT FAR* lprcUpdate; /* address of structure for update rect. */
```

The ScrollDC function scrolls a rectangle of bits horizontally and vertically.

| Parameter  | Description                                                                                                                                                                                                                                                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hdc        | Identifies the device context that contains the bits to be scrolled.                                                                                                                                                                                                                                                                     |
| dx         | Specifies the number of horizontal scroll units.                                                                                                                                                                                                                                                                                         |
| dy         | Specifies the number of vertical scroll units.                                                                                                                                                                                                                                                                                           |
| lprcScroll | Points to the <u>RECT</u> structure that contains the coordinates of the scrolling rectangle.                                                                                                                                                                                                                                            |
| lprcClip   | Points to the <u>RECT</u> structure that contains the coordinates of the clipping rectangle. When this rectangle is smaller than the original one pointed to by the lprcScroll parameter, scrolling occurs only in the smaller rectangle.                                                                                                |
| hrgnUpdate | Identifies the region uncovered by the scrolling process. The ScrollDC function defines this region; it is not necessarily a rectangle.                                                                                                                                                                                                  |
| lprcUpdate | Points to the <u>RECT</u> structure that receives the coordinates of the rectangle that bounds the scrolling update region. This is the largest rectangular area that requires repainting. The values in the structure when the function returns are in client coordinates, regardless of the mapping mode for the given device context. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

If the lprcUpdate parameter is NULL, Windows does not compute the update rectangle. If both the hrgnUpdate and lprcUpdate parameters are NULL, Windows does not compute the update region. If hrgnUpdate is not NULL, Windows assumes that it contains a valid handle of the region uncovered by the scrolling process (defined by the ScrollDC function).

When the ScrollDC function returns, the values in the structure pointed to by the lprcUpdate parameter are in client coordinates. This allows applications to use the update region in a call to the InvalidateRgn function, if required.

An application should use the ScrollWindow function when it is necessary to scroll the entire client area of a window; otherwise, it should use ScrollDC.

### See Also

InvalidateRgn, ScrollWindow, ScrollWindowEx, RECT



## ScrollWindow (2.x)

```
void ScrollWindow(hwnd, dx, dy, lprcScroll, lprcClip)
HWND hwnd;          /* handle of window to scroll */
int dx;             /* amount of horizontal scrolling */
int dy;             /* amount of vertical scrolling */
const RECT FAR* lprcScroll; /* address of structure with scroll rect. */
const RECT FAR* lprcClip;  /* address of structure with clip rect. */
```

The ScrollWindow function scrolls the contents of a window's client area.

| Parameter  | Description                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd       | Identifies the window to be scrolled.                                                                                                                                                                                                                                                                                                                                                                       |
| dx         | Specifies the amount, in device units, of horizontal scrolling. This parameter must be a negative value to scroll to the left.                                                                                                                                                                                                                                                                              |
| dy         | Specifies the amount, in device units, of vertical scrolling. This parameter must be a negative value to scroll up.                                                                                                                                                                                                                                                                                         |
| lprcScroll | Points to a <u>RECT</u> structure that specifies the portion of the client area to be scrolled. If this parameter is <u>NULL</u> , the entire client area is scrolled. The caret is repositioned if the cursor rectangle intersects the scroll rectangle.                                                                                                                                                   |
| lprcClip   | Points to a <u>RECT</u> structure that specifies the clipping rectangle to scroll. This structure takes precedence over the rectangle pointed to by the lprcScroll parameter. Only bits inside this rectangle are scrolled. Bits outside this rectangle are not affected even if they are in the lprcScroll rectangle. If this parameter is <u>NULL</u> , no clipping is performed on the scroll rectangle. |

### Returns

This function does not return a value.

### Comments

If the caret is in the window being scrolled, ScrollWindow automatically hides the caret to prevent it from being erased, then restores the caret after the scroll is finished. The caret position is adjusted accordingly if the caret rectangle intersects the scroll rectangle.

The area uncovered by the ScrollWindow function is not repainted, but it is combined into the window's update region. The application will eventually receive a WM\_PAINT message notifying it that the region needs repainting. To repaint the uncovered area at the same time the scrolling is done, call the UpdateWindow function immediately after calling ScrollWindow.

If the lprcScroll parameter is NULL, the positions of any child windows in the window are offset by the amount specified by the dx and dy parameters, and any invalid (unpainted) areas in the window are also offset. ScrollWindow is faster when lprcScroll is NULL.

If the lprcScroll parameter is not NULL, the positions of child windows are not changed and invalid areas in the window are not offset. To prevent updating problems when lprcScroll is not NULL, call the UpdateWindow function to repaint the window before calling ScrollWindow.

### See Also

ScrollDC, ScrollWindowEx, UpdateWindow, RECT

### ScrollWindowEx (3.1)

```
int ScrollWindowEx(hwnd, dx, dy, lprcScroll, lprcClip, hrgnUpdate, lprcUpdate, fuScroll)
HWND hwnd; /* handle of window to scroll */
int dx; /* amount of horizontal scrolling */
int dy; /* amount of vertical scrolling */
const RECT FAR* lprcScroll; /* address of structure with scroll rect. */
const RECT FAR* lprcClip; /* address of structure with clip rect. */
HRGN hrgnUpdate; /* handle of update region */
RECT FAR* lprcUpdate; /* address of structure for update rect. */
UINT fuScroll; /* scrolling flags */
```

The ScrollWindowEx function scrolls the contents of a window's client area. This function is similar to the ScrollWindow function, with some additional features.

| Parameter                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |                 |                                                                                                                                         |                      |                                                                                |                          |                                                                                                                                                                                                                                                                     |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------|----------------------|--------------------------------------------------------------------------------|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd                     | Identifies the window to be scrolled.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                 |                                                                                                                                         |                      |                                                                                |                          |                                                                                                                                                                                                                                                                     |
| dx                       | Specifies the amount, in device units, of horizontal scrolling. This parameter must be a negative value to scroll to the left.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                 |                                                                                                                                         |                      |                                                                                |                          |                                                                                                                                                                                                                                                                     |
| dy                       | Specifies the amount, in device units, of vertical scrolling. This parameter must be a negative value to scroll up.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                 |                                                                                                                                         |                      |                                                                                |                          |                                                                                                                                                                                                                                                                     |
| lprcScroll               | Points to a <u>RECT</u> structure that specifies the portion of the client area to be scrolled. If this parameter is NULL, the entire client area is scrolled.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                 |                                                                                                                                         |                      |                                                                                |                          |                                                                                                                                                                                                                                                                     |
| lprcClip                 | Points to a <u>RECT</u> structure that specifies the clipping rectangle to scroll. This structure takes precedence over the rectangle pointed to by the lprcScroll parameter. Only bits inside this rectangle are scrolled. Bits outside this rectangle are not affected even if they are in the lprcScroll rectangle. If this parameter is NULL, no clipping is performed on the scroll rectangle.                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                 |                                                                                                                                         |                      |                                                                                |                          |                                                                                                                                                                                                                                                                     |
| hrgnUpdate               | Identifies the region that is modified to hold the region invalidated by scrolling. This parameter may be NULL.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                 |                                                                                                                                         |                      |                                                                                |                          |                                                                                                                                                                                                                                                                     |
| lprcUpdate               | Points to a <u>RECT</u> structure that will receive the boundaries of the rectangle invalidated by scrolling. This parameter may be NULL.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |                 |                                                                                                                                         |                      |                                                                                |                          |                                                                                                                                                                                                                                                                     |
| fuScroll                 | Specifies flags that control scrolling. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><u>SW_ERASE</u></td><td>When specified with <u>SW_INVALIDATE</u>, erases the newly invalidated region by sending a <u>WM_ERASEBKGND</u> message to the window.</td></tr><tr><td><u>SW_INVALIDATE</u></td><td>Invalidates the region identified by the hrgnUpdate parameter after scrolling.</td></tr><tr><td><u>SW_SCROLLCHILDREN</u></td><td>Scrolls all child windows that intersect the rectangle pointed to by lprcScroll by the number of pixels specified in the dx and dy parameters. Windows sends a <u>WM_MOVE</u> message to all child windows that intersect lprcScroll, even if they do not move. The</td></tr></tbody></table> | Value | Meaning | <u>SW_ERASE</u> | When specified with <u>SW_INVALIDATE</u> , erases the newly invalidated region by sending a <u>WM_ERASEBKGND</u> message to the window. | <u>SW_INVALIDATE</u> | Invalidates the region identified by the hrgnUpdate parameter after scrolling. | <u>SW_SCROLLCHILDREN</u> | Scrolls all child windows that intersect the rectangle pointed to by lprcScroll by the number of pixels specified in the dx and dy parameters. Windows sends a <u>WM_MOVE</u> message to all child windows that intersect lprcScroll, even if they do not move. The |
| Value                    | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |                 |                                                                                                                                         |                      |                                                                                |                          |                                                                                                                                                                                                                                                                     |
| <u>SW_ERASE</u>          | When specified with <u>SW_INVALIDATE</u> , erases the newly invalidated region by sending a <u>WM_ERASEBKGND</u> message to the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |                 |                                                                                                                                         |                      |                                                                                |                          |                                                                                                                                                                                                                                                                     |
| <u>SW_INVALIDATE</u>     | Invalidates the region identified by the hrgnUpdate parameter after scrolling.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                 |                                                                                                                                         |                      |                                                                                |                          |                                                                                                                                                                                                                                                                     |
| <u>SW_SCROLLCHILDREN</u> | Scrolls all child windows that intersect the rectangle pointed to by lprcScroll by the number of pixels specified in the dx and dy parameters. Windows sends a <u>WM_MOVE</u> message to all child windows that intersect lprcScroll, even if they do not move. The                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                 |                                                                                                                                         |                      |                                                                                |                          |                                                                                                                                                                                                                                                                     |

caret is repositioned when a child window is scrolled and the cursor rectangle intersects the scroll rectangle.

#### Returns

The return value is `SIMPLEREGION` (rectangular invalidated region), `COMPLEXREGION` (nonrectangular invalidated region; overlapping rectangles), or `NULLREGION` (no invalidated region), if the function is successful. Otherwise, the return value is `ERROR`.

#### Comments

If `SW_INVALIDATE` and `SW_ERASE` are not specified, `ScrollWindowEx` does not invalidate the area that is scrolled away from. If either of these flags is set, `ScrollWindowEx` invalidates this area. The area is not updated until the application calls the `UpdateWindow` function, calls the `RedrawWindow` function (specifying `RDW_UPDATENOW` or `RDW_ERASENOW`), or retrieves the `WM_PAINT` message from the application queue.

If the window has the `WS_CLIPCHILDREN` style, the returned areas specified by `hrgnUpdate` and `lprcUpdate` represent the total area of the scrolled window that must be updated, including any areas in child windows that need updating.

If the `SW_SCROLLCHILDREN` flag is specified, Windows will not properly update the screen if part of a child window is scrolled. The part of the scrolled child window that lies outside the source rectangle will not be erased and will not be redrawn properly in its new destination. Use the `DeferWindowPos` function to move child windows that do not lie completely within the `lprcScroll` rectangle. The cursor is repositioned if the `SW_SCROLLCHILDREN` flag is set and the caret rectangle intersects the scroll rectangle.

All input and output coordinates (for `lprcScroll`, `lprcClip`, `lprcUpdate`, and `hrgnUpdate`) are assumed to be in client coordinates, regardless of whether the window has the `CS_OWNDC` or `CS_CLASSDC` class style. Use the `LPToDP` and `DPToLP` functions to convert to and from logical coordinates, if necessary.

#### See Also

[RedrawWindow](#), [ScrollDC](#), [ScrollWindow](#), [UpdateWindow](#), [RECT](#)

SW\_ERASE 0x0004

When specified with SW\_INVALIDATE, erases the newly invalidated region by sending a WM\_ERASEBKGND message to the window.

SW\_ERASE 0x0004

SW\_INVALIDATE 0x0002

Invalidates the region identified by the hrgnUpdate parameter after scrolling.

SW\_INVALIDATE 0x0002

SW\_SCROLLCHILDREN 0x0001

Scrolls all child windows that intersect the rectangle pointed to by `lprcScroll` by the number of pixels specified in the `dx` and `dy` parameters. Windows sends a WM\_MOVE message to all child windows that intersect `lprcScroll`, even if they do not move. The caret is repositioned when a child window is scrolled and the cursor rectangle intersects the scroll rectangle.



SW\_SCROLLCHILDREN 0x0001

## SelectPalette (3.0)

```
HPALETTE SelectPalette(hdc, hpal, fPalBack)
HDC hdc; /* handle of device context */
HPALETTE hpal; /* handle of palette */
/
BOOL fPalBack;
* flag for forcing palette to background
/
```

The SelectPalette function selects the specified logical palette into the given device context. The selected palette replaces the previous palette for that device context.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hdc       | Identifies the device context.                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| hpal      | Identifies the logical palette to be selected.                                                                                                                                                                                                                                                                                                                                                                                                                         |
| fPalBack  | Specifies whether the logical palette is always to be a background palette. If this parameter is nonzero, the selected palette is always a background palette. If this parameter is zero and the device context is attached to a window, the logical palette is a foreground palette when the window has the input focus. (The device context is attached to a window if it was obtained by using the <u>GetDC</u> function or if the window-class style is CS_OWNDC.) |

### Returns

The return value is the handle of the previous logical palette for the given device context, if the function is successful. Otherwise, it is NULL.

### Comments

An application can select a logical palette into more than one device context. However, changes to a logical palette will affect all device contexts for which it is selected. If an application selects a palette into more than one device context, the device contexts must all belong to the same physical device.

### Example

The following example calls the SelectPalette function to select a logical palette into a device context and then calls the RealizePalette function to change the palette size:

```
HPALETTE hpal, hPalPrevious;

hdc = GetDC(hwnd);

hPalPrevious = SelectPalette(hdc, hpal, FALSE);
if (RealizePalette(hdc) == NULL)
    MessageBox(hwnd, "Can't realize palette", "Error", MB_OK);

ReleaseDC(hwnd, hdc);
```

### See Also

CreatePalette, GetDC, RealizePalette

## SendDlgItemMessage (2.x)

```
LRESULT SendDlgItemMessage(HWND hDlg, int idDlgItem, UINT uMsg, WPARAM wParam, LPARAM lParam)
/* handle of dialog box */
/* identifier of dialog box item */
/* message */
/* first message parameter */
/* second message parameter */
```

The SendDlgItemMessage function sends a message to a control in a dialog box.

| Parameter | Description                                                                |
|-----------|----------------------------------------------------------------------------|
| hWndDlg   | Identifies the dialog box that contains the control.                       |
| idDlgItem | Specifies the identifier of the dialog item that will receive the message. |
| uMsg      | Specifies the message to be sent.                                          |
| wParam    | Specifies 16 bits of additional message-dependent information.             |
| lParam    | Specifies 32 bits of additional message-dependent information.             |

### Returns

The return value specifies the result of the message processing and depends on the message sent.

### Comments

The SendDlgItemMessage function does not return until the message has been processed.

Using SendDlgItemMessage is identical to retrieving a handle of the given control and calling the SendMessage function.

### See Also

PostMessage, SendMessage

### SendDriverMessage (3.1)

```
LRESULT SendDriverMessage(hdrvr, msg, lParam1, lParam2)
HDRVR hdrvr;          /* handle of installable driver */
UINT msg;             /* message */
LPARAM lParam1;
/* first message parameter
/
LPARAM lParam2;
/* second message parameter
/
```

The SendDriverMessage function sends the specified message to the given installable driver.

| Parameter | Description                                                                                                                                                                 |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hdrvr     | Identifies the installable driver.                                                                                                                                          |
| msg       | Specifies the message that the driver must process. The following messages should never be sent by an application directly to the driver; they are sent only by the system: |

DRV\_CLOSE  
DRV\_DISABLE  
DRV\_ENABLE  
DRV\_EXITAPPLICATION  
DRV\_EXITSESSION  
DRV\_FREE  
DRV\_LOAD  
DRV\_OPEN

|         |                                                                |
|---------|----------------------------------------------------------------|
| lParam1 | Specifies 32 bits of additional message-dependent information. |
| lParam2 | Specifies 32 bits of additional message-dependent information. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### See Also

DefDriverProc

## SendMessage (2.x)

```
LRESULT SendMessage(hwnd, uMsg, wParam, lParam)
HWND hwnd;          /* handle of destination window */
UINT uMsg;          /* message to send */
/
WPARAM wParam;
/* first message parameter
/
LPARAM lParam;
/* second message parameter
/
```

The SendMessage function sends the specified message to the given window or windows. The function calls the window procedure for the window and does not return until that window procedure has processed the message. This is in contrast to the PostMessage function, which places (posts) the message in the window's message queue and returns immediately.

| Parameter | Description                                                                                                                                                                                                     |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window to which the message will be sent. If this parameter is <code>HWND_BROADCAST</code> , the message will be sent to all top-level windows, including disabled or invisible unowned windows. |
| uMsg      | Specifies the message to be sent.                                                                                                                                                                               |
| wParam    | Specifies 16 bits of additional message-dependent information.                                                                                                                                                  |
| lParam    | Specifies 32 bits of additional message-dependent information.                                                                                                                                                  |

### Returns

The return value specifies the result of the message processing and depends on the message sent.

### Comments

If the message is being sent to another application and the wParam or lParam parameter is used to pass a handle or pointer to global memory, the memory should be allocated by the GlobalAlloc function using the GMEM\_SHARE flag.

### Example

The following example calls the SendMessage function to send an EM\_SETSEL message to a multiline edit control, telling it to select all the text. It then calls SendMessage to send a WM\_COPY message to copy the selected text to the clipboard.

```
SendMessage(hwndMle, EM_SETSEL, 0, MAKELONG(0, -1));
SendMessage(hwndMle, WM_COPY, 0, 0L);
```

### See Also

InSendMessage, PostMessage, SendDlgItemMessage

### Windows 3.1 changes

In Windows 3.0, specifying a window handle of -1 would not send the message to disabled or invisible unowned windows. In Windows 3.1, if a window handle of `HWND_BROADCAST (-1)` is specified, the message is sent to disabled or invisible unowned windows.

## SetActiveWindow (2.x)

HWND SetActiveWindow(HWND)

HWND hwnd; /\* handle of window to activate \*/

The SetActiveWindow function makes the specified top-level window the active window.

| Parameter | Description                                      |
|-----------|--------------------------------------------------|
| hwnd      | Identifies the top-level window to be activated. |

### Returns

The return value identifies the window that was previously active, if the function is successful.

### Comments

The SetActiveWindow function should be used with care, since it allows an application to arbitrarily take over the active window and input focus. Normally, Windows takes care of all activation.

### See Also

[GetActiveWindow](#), [SetCapture](#), [SetFocus](#)

## SetCapture (2.x)

HWND SetCapture(HWND)

HWND hwnd; /\* handle of window to receive all mouse messages \*/

The SetCapture function sets the mouse capture to the specified window. With the mouse capture set to a window, all mouse input is directed to that window, regardless of whether the cursor is over that window. Only one window can have the mouse capture at a time.

| Parameter | Description                                                  |
|-----------|--------------------------------------------------------------|
| hwnd      | Identifies the window that is to receive all mouse messages. |

### Returns

The return value is the handle of the window that previously received all mouse input, if the function is successful. It is NULL if there is no such window.

### Comments

When the window no longer requires all mouse input, the application should call the ReleaseCapture function so that other windows can receive mouse input.

### See Also

ReleaseCapture



## SetCaretBlinkTime (2.x)

```
void SetCaretBlinkTime(uMSeconds)
UINT uMSeconds;      /* blink rate in milliseconds */
```

The SetCaretBlinkTime function sets the caret blink rate. The blink rate is the elapsed time, in milliseconds, between caret flashes.

| Parameter | Description                                    |
|-----------|------------------------------------------------|
| uMSeconds | Specifies the new blink rate, in milliseconds. |

### Returns

This function does not return a value.

### Comments

The caret flashes on or off every uMSeconds milliseconds. One complete flash (off-on) takes twice uMSeconds milliseconds.

The caret is a shared resource. A window should set the caret blink rate only if it owns the caret. It should restore the previous rate before it loses the input focus or becomes inactive.

### See Also

[GetCaretBlinkTime](#)

## SetCaretPos (2.x)

```
void SetCaretPos(x, y)
int x;    /* horizontal position */
int y;    /* vertical position   */
```

The SetCaretPos function sets the position of the caret.

| Parameter | Description                                                          |
|-----------|----------------------------------------------------------------------|
| x         | Specifies the new x-coordinate, in client coordinates, of the caret. |
| y         | Specifies the new y-coordinate, in client coordinates, of the caret. |

### Returns

This function does not return a value.

### Comments

The SetCaretPos function moves the caret only if it is owned by a window in the current task. SetCaretPos moves the caret whether or not the caret is hidden.

The caret is a shared resource. A window should not move the caret if it does not own the caret.

### See Also

[GetCaretPos](#)

## SetClassLong (2.x)

```
LONG SetClassLong(hwnd, nIndex, nVal)
HWND hwnd;      /* handle of window      */
int nIndex;     /* index of value to change */
/
LONG nVal;
* new value
/
```

The SetClassLong function sets a long value at the specified offset into the extra class memory for the window class to which the specified window belongs. Extra class memory is reserved by specifying a nonzero value in the cbClsExtra member of the WNDCLASS structure used with the RegisterClass function.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window.                                                                                                                                                                                                                                                                                                                                                                                  |
| nIndex    | Specifies the zero-based byte offset of the long value to change. Valid values are in the range zero through the number of bytes of class memory, minus four. (For example, if 12 or more bytes of extra class memory were specified, a value of 8 would be an index to the third long integer.) This parameter can also be <u>GCL_WNDPROC</u> , which sets a new long pointer to the window procedure. |
| nVal      | Specifies the replacement value.                                                                                                                                                                                                                                                                                                                                                                        |

### Returns

The return value is the previous value of the specified long integer, if the function is successful. Otherwise, it is zero.

### Comments

If the SetClassLong function and GCL\_WNDPROC index are used to set a window procedure, the specified window procedure must have the window-procedure form and be exported in the module-definition file. For more information, see the description of the RegisterClass function.

Calling SetClassLong with the GCL\_WNDPROC index creates a subclass of the window class that affects all windows subsequently created by using the class.

Applications should not call SetClassLong with the GCL\_MENUNAME value.

To access any extra 4-byte values allocated when the window-class structure was created, use a positive byte offset as the index specified by the nIndex parameter, starting at 0 for the first 4-byte value in the extra space, 4 for the next 4-byte value, and so on.

### See Also

GetClassLong, RegisterClass, SetClassWord, WNDCLASS

## SetClassWord (2.x)

```
WORD SetClassWord(hwnd, nIndex, wNewWord)
HWND hwnd;          /* handle of window */
int nIndex;         /* index of value to change */
/
WORD wNewWord;
* new value
/
```

The SetClassWord function sets a word value at the specified offset into the extra class memory for the window class to which the given window belongs. Extra class memory is reserved by specifying a nonzero value in the cbClsExtra member of the WNDCLASS structure used with the RegisterClass function.

| Parameter         | Description                                                                                                                                                                                                                                                                                                                                                                                 |       |         |                   |                                          |             |                                |           |                               |           |                                            |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|-------------------|------------------------------------------|-------------|--------------------------------|-----------|-------------------------------|-----------|--------------------------------------------|
| hwnd              | Identifies the window.                                                                                                                                                                                                                                                                                                                                                                      |       |         |                   |                                          |             |                                |           |                               |           |                                            |
| nIndex            | Specifies the zero-based byte offset of the word value to change. Valid values are in the range zero through the number of bytes of class memory, minus two (for example, if 10 or more bytes of extra class memory were specified, a value of 8 would be an index to the fifth integer), or one of the following values:                                                                   |       |         |                   |                                          |             |                                |           |                               |           |                                            |
|                   | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>GCW_HBRBACKGROUND</td><td>Sets a new handle of a background brush.</td></tr><tr><td>GCW_HCURSOR</td><td>Sets a new handle of a cursor.</td></tr><tr><td>GCW_HICON</td><td>Sets a new handle of an icon.</td></tr><tr><td>GCW_STYLE</td><td>Sets a new style bit for the window class.</td></tr></tbody></table> | Value | Meaning | GCW_HBRBACKGROUND | Sets a new handle of a background brush. | GCW_HCURSOR | Sets a new handle of a cursor. | GCW_HICON | Sets a new handle of an icon. | GCW_STYLE | Sets a new style bit for the window class. |
| Value             | Meaning                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                   |                                          |             |                                |           |                               |           |                                            |
| GCW_HBRBACKGROUND | Sets a new handle of a background brush.                                                                                                                                                                                                                                                                                                                                                    |       |         |                   |                                          |             |                                |           |                               |           |                                            |
| GCW_HCURSOR       | Sets a new handle of a cursor.                                                                                                                                                                                                                                                                                                                                                              |       |         |                   |                                          |             |                                |           |                               |           |                                            |
| GCW_HICON         | Sets a new handle of an icon.                                                                                                                                                                                                                                                                                                                                                               |       |         |                   |                                          |             |                                |           |                               |           |                                            |
| GCW_STYLE         | Sets a new style bit for the window class.                                                                                                                                                                                                                                                                                                                                                  |       |         |                   |                                          |             |                                |           |                               |           |                                            |
| wNewWord          | Specifies the replacement value.                                                                                                                                                                                                                                                                                                                                                            |       |         |                   |                                          |             |                                |           |                               |           |                                            |

### Returns

The return value is the previous value of the specified word, if the function is successful. Otherwise, it is zero.

### Comments

The SetClassWord function should be used with care. For example, it is possible to change the background color for a class by using SetClassWord, but this change does not cause all windows belonging to the class to be repainted immediately. Applications should not attempt to set the class word values of any class attribute except those listed for the nIndex parameter.

To access any extra 2-byte values allocated when the window-class structure was created, use a positive byte offset as the index specified by the nIndex parameter, starting at 0 for the first 2-byte value in the extra space, 2 for the next 2-byte value, and so on.

### See Also

GetClassWord, RegisterClass, SetClassLong, WNDCLASS

## SetClipboardData (2.x)

```
HANDLE SetClipboardData(uFormat, hData)
UINT uFormat;          /* clipboard format */
HANDLE hData;          /* data handle */
```

The SetClipboardData function sets the data in the clipboard. The application must have called the OpenClipboard function before calling the SetClipboardData function.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uFormat   | Specifies the format of the data. It can be any one of the system-defined formats or a format registered by the <u>RegisterClipboardFormat</u> function. For a list of system-defined formats, see the following <u>Comments</u> section.                                                                                                                                                                                                                                                                                                                                   |
| hData     | Identifies the data to be placed in the clipboard. For all formats except CF_BITMAP and CF_PALETTE, this parameter must be a handle of the memory allocated by the <u>GlobalAlloc</u> function. For CF_BITMAP format, the hData parameter is a bitmap handle (see the description of the <u>LoadBitmap</u> function). For the CF_PALETTE format, hData is a palette handle (see the description of the <u>CreatePalette</u> function).<br>If this parameter is NULL, the owner of the clipboard will be sent a <u>WM_RENDERFORMAT</u> message when it must supply the data. |

### Returns

The return value is a handle of the data, if the function is successful. Otherwise, it is NULL.

### Comments

If the hData parameter contains a handle of the memory allocated by the GlobalAlloc function, the application must not use this handle once it has called the SetClipboardData function.

Following are the system-defined clipboard formats:

| Value              | Meaning                                                                                                                                                                             |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CF_BITMAP          | The data is a bitmap.                                                                                                                                                               |
| CF_DIB             | The data is a memory object containing a <u>BITMAPINFO</u> structure followed by the bitmap data.                                                                                   |
| CF_DIF             | The data is in Data Interchange Format (DIF).                                                                                                                                       |
| CF_DSPBITMAP       | The data is a bitmap representation of a private format. This data is displayed in bitmap format in lieu of the privately formatted data.                                           |
| CF_DSPMETAFILEPICT | The data is a metafile representation of a private data format. This data is displayed in metafile-picture format in lieu of the privately formatted data.                          |
| CF_DSPTEXT         | The data is a textual representation of a private data format. This data is displayed in text format in lieu of the privately formatted data.                                       |
| CF_METAFILEPICT    | The data is a metafile (see the description of the <u>METAFILEPICT</u> structure).                                                                                                  |
| CF_OEMTEXT         | The data is an array of text characters in the OEM character set. Each line ends with a carriage return–linefeed (CR-LF) combination. A null character signals the end of the data. |
| CF_OWNERDISPLAY    | The data is in a private format that the clipboard owner must display.                                                                                                              |
| CF_PALETTE         | The data is a color palette.                                                                                                                                                        |
| CF_PENDATA         | The data is for the pen extensions to the Windows operating system.                                                                                                                 |
| CF_RIFF            | The data is in Resource Interchange File Format (RIFF).                                                                                                                             |
| CF_SYLK            | The data is in Microsoft Symbolic Link (SYLK) format.                                                                                                                               |
| CF_TEXT            | The data is an array of text characters. Each line ends with a carriage return–linefeed (CR-LF) combination. A null character signals the end of the data.                          |
| CF_TIFF            | The data is in Tag Image File Format (TIFF).                                                                                                                                        |
| CF_WAVE            | The data describes a sound wave. This is a subset of the CF_RIFF data format; it can be used only for RIFF WAVE files.                                                              |

Private data formats in the range CF\_PRIVATEFIRST through CF\_PRIVATELAST are not automatically freed when the data is removed from the clipboard. Data handles associated with these formats should be freed upon receiving a WM\_DESTROYCLIPBOARD message.

Private data formats in the range CF\_GDIOBJFIRST through CF\_GDIOBJLAST will be automatically removed by a call to the DeleteObject function when the data is removed from the clipboard.

If Windows Clipboard is running, it will not update its window to show the data placed in the clipboard by the SetClipboardData until after the CloseClipboard function is called.

See Also

CloseClipboard, GlobalAlloc, OpenClipboard, GetClipboardData, RegisterClipboardFormat, BITMAPINFO, WM\_RENDERFORMAT

## SetClipboardViewer (2.x)

```
HWND SetClipboardViewer(HWND  
HWND hwnd; /* handle of clipboard viewer */
```

The `SetClipboardViewer` function adds the given window to the chain of windows that are notified (by means of the WM\_DRAWCLIPBOARD message) whenever the contents of the clipboard are changed.

| Parameter         | Description                                                       |
|-------------------|-------------------------------------------------------------------|
| <code>hwnd</code> | Identifies the window to receive clipboard-viewer chain messages. |

### Returns

The return value is the handle of the next window in the clipboard-viewer chain, if the function is successful.

### Comments

Applications should save this handle in static memory and use it when responding to clipboard-viewer chain messages.

Windows that are part of the clipboard-viewer chain must respond to WM\_CHANGECHAIN, WM\_DRAWCLIPBOARD, and WM\_DESTROY messages.

To remove itself from the clipboard-viewer chain, an application must call the ChangeClipboardChain function.

### See Also

ChangeClipboardChain, GetClipboardViewer, WM\_CHANGECHAIN, WM\_DESTROY, WM\_DRAWCLIPBOARD

## SetCommBreak (2.x)

```
int SetCommBreak(idComDev)
int idComDev;    /* device to suspend */
```

The SetCommBreak function suspends character transmission and places the communications device in a break state.

| Parameter | Description                                                                                           |
|-----------|-------------------------------------------------------------------------------------------------------|
| idComDev  | Specifies the communications device to be suspended. The <u>OpenComm</u> function returns this value. |

### Returns

The return value is zero if the function is successful. Otherwise, it is less than zero.

### Comments

The communications device remains suspended until the application calls the ClearCommBreak function.

### See Also

ClearCommBreak, OpenComm



## SetCommEventMask (2.x)

```
UINT FAR* SetCommEventMask(idComDev, fuEvtMask)
int idComDev; /* device to enable */
UINT fuEvtMask; /* events to enable */
```

The SetCommEventMask function enables events in the event word of the specified communications device.

| Parameter | Description                                                                                              |
|-----------|----------------------------------------------------------------------------------------------------------|
| idComDev  | Specifies the communications device to be enabled. The <u>OpenComm</u> function returns this value.      |
| fuEvtMask | Specifies which events are to be enabled. This parameter can be any combination of the following values: |

| Value             | Meaning                                                                                                                                     |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <u>EV_BREAK</u>   | Set when a break is detected on input.                                                                                                      |
| <u>EV_CTS</u>     | Set when the CTS (clear-to-send) signal changes state.                                                                                      |
| <u>EV_CTSS</u>    | Set to indicate the current state of the CTS signal.                                                                                        |
| <u>EV_DSR</u>     | Set when the DSR (data-set-ready) signal changes state.                                                                                     |
| <u>EV_ERR</u>     | Set when a line-status error occurs. Line-status errors are <u>CE_FRAME</u> , <u>CE_OVERRUN</u> , and <u>CE_RXPARITY</u> .                  |
| <u>EV_PERR</u>    | Set when a printer error is detected on a parallel device. Errors are <u>CE_DNS</u> , <u>CE_IOE</u> , <u>CE_LOOP</u> , and <u>CE_PTO</u> .  |
| <u>EV_RING</u>    | Set to indicate the state of ring indicator during the last modem interrupt.                                                                |
| <u>EV_RLSD</u>    | Set when the RLSD (receive-line-signal-detect) signal changes state.                                                                        |
| <u>EV_RLSDS</u>   | Set to indicate the current state of the RLSD signal.                                                                                       |
| <u>EV_RXCHAR</u>  | Set when any character is received and placed in the receiving queue.                                                                       |
| <u>EV_RXFLAG</u>  | Set when the event character is received and placed in the receiving queue. The event character is specified in the device's control block. |
| <u>EV_TXEMPTY</u> | Set when the last character in the transmission queue is sent.                                                                              |

### Returns

The return value is a pointer to the event word for the specified communications device, if the function is successful. Each bit in the event word specifies whether a given event has occurred. A bit is 1 if the event has occurred.

### Comments

Only enabled events are recorded. The GetCommEventMask function retrieves and clears the event word.

### See Also

GetCommEventMask, OpenComm

EV\_BREAK 0x0040

Set when a break is detected on input.

EV\_BREAK 0x0040

EV\_CTS 0x0008

Set when the CTS (clear-to-send) signal changes state.

EV\_CTS 0x0008

EV\_CTSS 0x0400

Set to indicate the current state of the CTS signal.

EV\_CTSS 0x0400

EV\_DSR 0x0010

Set when the DSR (data-set-ready) signal changes state.



EV\_DSR 0x0010

EV\_ERR 0x0080

Set when a line-status error occurs. Line-status errors are CE\_FRAME, CE\_OVERRUN, and CE\_RXPARITY.

EV\_ERR 0x0080

EV\_PERR 0x0200

Set when a printer error is detected on a parallel device. Errors are CE\_DNS, CE\_IOE, CE\_LOOP, and CE\_PTO.

EV\_PERR 0x0200

EV\_RING 0x0100

Set to indicate the state of ring indicator during the last modem interrupt.

EV\_RING 0x0100

EV\_RLSD 0x0020

Set when the RLSD (receive-line-signal-detect) signal changes state.



EV\_RLSD 0x0020

EV\_RLSDS 0x1000

Set to indicate the current state of the RLSD signal.

EV\_RLSDS 0x1000

EV\_RXCHAR 0x0001

Set when any character is received and placed in the receiving queue.

EV\_RXCHAR 0x0001

EV\_RXFLAG 0x0002

Set when the event character is received and placed in the receiving queue. The event character is specified in the device's control block.

EV\_RXFLAG 0x0002

EV\_TXEMPTY 0x0004

Set when the last character in the transmission queue is sent.



EV\_TXEMPTY 0x0004

## SetCommState (2.x)

```
int SetCommState(lpdcb)
const DCB FAR* lpdcb;          /* address of device control block */
```

The SetCommState function sets a communications device to the state specified by a device control block.

| Parameter | Description                                                                                                                                                            |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpdcb     | Points to a <u>DCB</u> structure that contains the desired communications settings for the device. The <u>Id</u> member of the DCB structure must identify the device. |

### Returns

The return value is zero if the function is successful. Otherwise, it is less than zero.

### Example

The following example uses the BuildCommDCB and SetCommState functions to set up COM1 at 9600 baud, no parity, 8 data bits, and 1 stop bit:

```
idComDev = OpenComm("COM1", 1024, 128);
if (idComDev < 0) {
    ShowError(idComDev, "OpenComm");
    return 0;
}

err = BuildCommDCB("COM1:9600,n,8,1", &dcb);
if (err < 0) {
    ShowError(err, "BuildCommDCB");
    return 0;
}

err = SetCommState(&dcb);
if (err < 0) {
    ShowError(err, "SetCommState");
    return 0;
}
```

### Comments

This function reinitializes all hardware and controls as defined by the DCB structure, but it does not empty transmission or receiving queues.

### See Also

GetCommState, DCB

## SetCursor (2.x)

```
HCURSOR SetCursor(hcur)
HCURSOR hcur;      /* handle of cursor */
```

The SetCursor function changes the given cursor.

| Parameter | Description                                                                                                                                                                   |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hcur      | Identifies the cursor resource. The resource must have been loaded by using the <u>LoadCursor</u> function. If this parameter is NULL, the cursor is removed from the screen. |

### Returns

The return value is the handle of the previous cursor, if the function is successful. It is NULL if there is no previous cursor.

### Comments

The cursor is set only if the new cursor is different from the previous cursor; otherwise, the function returns immediately. The function is quite fast if the new cursor is the same as the old.

The cursor is a shared resource. A window should set the cursor only when the cursor is in the window's client area or when the window is capturing all mouse input. In systems without a mouse, the window should restore the previous cursor before the cursor leaves the client area or before the window relinquishes control to another window.

Any application that must set the cursor while it is in a window must ensure that the class cursor for the given window's class is set to NULL. If the class cursor is not NULL, the system restores the previous shape each time the mouse is moved.

### Example

The following example sets the hourglass cursor during a time-consuming operation and restores the cursor afterward:

```
HCURSOR hcurSave;

/* Set the cursor to the hourglass and save the previous cursor. */
hcurSave = SetCursor(LoadCursor(NULL, IDC_WAIT));

    .
    . /* Perform some time-consuming operation */
    .

/* Restore the previous cursor. */

SetCursor(hcurSave);
```

### See Also

GetCursor, LoadCursor, ShowCursor, WM\_SETCURSOR

## SetCursorPos (2.x)

```
void SetCursorPos(x, y)
int x;      /* horizontal position */
int y;      /* vertical position   */
```

The SetCursorPos function sets the position, in screen coordinates, of the cursor. If the new coordinates are not within the screen rectangle set by the most recent ClipCursor function, Windows automatically adjusts the coordinates so that the cursor stays within the rectangle.

| Parameter | Description                                                           |
|-----------|-----------------------------------------------------------------------|
| x         | Specifies the new x-coordinate, in screen coordinates, of the cursor. |
| y         | Specifies the new y-coordinate, in screen coordinates, of the cursor. |

### Returns

This function does not return a value.

### Comments

The cursor is a shared resource. A window should move the cursor only when the cursor is in its client area.

### See Also

ClipCursor, GetCursorPos

## SetDlgItemInt (2.x)

```
void SetDlgItemInt(hwndDlg, idControl, uValue, fSigned)
HWND hwndDlg; /* handle of dialog box */
int idControl; /* identifier of control */
/
UINT uValue; /* value to set */
/
BOOL fSigned; /* signed or unsigned indicator */
/
```

The SetDlgItemInt function sets the text of a given control in a dialog box to the string representation of a specified integer value.

| Parameter | Description                                                                                                                                                                                                                                                                     |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwndDlg   | Identifies the dialog box that contains the control.                                                                                                                                                                                                                            |
| idControl | Specifies the control to be changed.                                                                                                                                                                                                                                            |
| uValue    | Specifies the integer value used to generate the item text.                                                                                                                                                                                                                     |
| fSigned   | Specifies whether the uValue parameter is signed or unsigned. If this parameter is TRUE, uValue is signed. If this parameter is TRUE and uValue is less than zero, a minus sign is placed before the first digit in the string. If this parameter is FALSE, uValue is unsigned. |

### Returns

This function does not return a value.

### Comments

SetDlgItemInt sends a WM\_SETTEXT message to the given control.

### See Also

GetDlgItemInt, SetDlgItemText, WM\_SETTEXT

## SetDlgItemText (2.x)

```
void SetDlgItemText(hwndDlg, idControl, lpsz)
HWND hwndDlg;    /* handle of dialog box */
int idControl;   /* identifier of control */
LPCSTR lpsz;     /* text to set */
/
```

The SetDlgItemText function sets the title or text of a control in a dialog box.

| Parameter | Description                                                                              |
|-----------|------------------------------------------------------------------------------------------|
| hwndDlg   | Identifies the dialog box that contains the control.                                     |
| idControl | Identifies the control whose text is to be set.                                          |
| lpsz      | Points to the null-terminated string that contains the text to be copied to the control. |

### Returns

This function does not return a value.

### Comments

The SetDlgItemText function sends a WM\_SETTEXT message to the given control.

### See Also

GetDlgItemText, SetDlgItemInt, WM\_SETTEXT

## SetDoubleClickTime (2.x)

```
void SetDoubleClickTime(uInterval)
UINT uInterval;          /* double-click interval */
```

The SetDoubleClickTime function sets the double-click time for the mouse. A double-click is a series of two clicks of the mouse button, the second occurring within a specified time after the first. The double-click time is the maximum number of milliseconds that may occur between the first and second clicks of a double-click.

| Parameter | Description                                                                |
|-----------|----------------------------------------------------------------------------|
| uInterval | Specifies the number of milliseconds that can occur between double-clicks. |

### Returns

This function does not return a value.

### Comments

If the uInterval parameter is zero, Windows uses the default double-click time of 500 milliseconds.

The SetDoubleClickTime function alters the double-click time for all windows in the system.

### See Also

[GetDoubleClickTime](#)

## SetFocus (2.x)

```
HWND SetFocus(HWND)
HWND hwnd; /* handle of window to receive focus */
```

The SetFocus function sets the input focus to the given window. All subsequent keyboard input is directed to this window. The window, if any, that previously had the input focus loses it.

| Parameter | Description                                                                                             |
|-----------|---------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window to receive the keyboard input. If this parameter is NULL, keystrokes are ignored. |

### Returns

The return value identifies the window that previously had the input focus, if the function is successful. It is NULL if there is no such window or if the specified handle is invalid.

### Comments

The SetFocus function sends a WM\_KILLFOCUS message to the window that loses the input focus and a WM\_SETFOCUS message to the window that receives the input focus. It also activates either the window that receives the focus or the parent of the window that receives the focus.

If a window is active but does not have the focus (that is, no window has the focus), any key pressed will produce the WM\_SYSCHAR, WM\_SYSKEYDOWN, or WM\_SYSKEYUP message. If the VK\_MENU key is also pressed, the lParam parameter of the message will have bit 30 set. Otherwise, the messages that are produced do not have this bit set.

### See Also

GetActiveWindow, GetFocus, SetActiveWindow, SetCapture, WM\_KILLFOCUS, WM\_SETFOCUS, WM\_SYSCHAR, WM\_SYSKEYDOWN, WM\_SYSKEYUP



## SetKeyboardState (2.x)

```
void SetKeyboardState(lpKeyState)
BYTE FAR* lpKeyState;          /* address of array with virtual-key codes */
```

The SetKeyboardState function copies a 256-byte array of keyboard key states into the Windows keyboard-state table.

| Parameter  | Description                                                   |
|------------|---------------------------------------------------------------|
| lpKeyState | Points to a 256-byte array that contains keyboard key states. |

**Returns**  
This function does not return a value.

**Comments**  
In many cases, an application should call the GetKeyboardState function first to initialize the 256-byte array. The application should then change the desired bytes.

SetKeyboardState sets the LEDs and BIOS flags for the NUMLOCK, CAPSLOCK, and SCROLL LOCK keys according to the toggle state of the VK\_NUMLOCK, VK\_CAPITAL, and VK\_SCROLL entries of the array.

For more information, see the description of the GetKeyboardState function.

**Example**  
The following example simulates the pressing of the CTRL key:

```
BYTE pbKeyState[256];

GetKeyboardState((LPBYTE) &pbKeyState);
pbKeyState[VK_CONTROL] |= 0x80;
SetKeyboardState((LPBYTE) &pbKeyState);
```

**See Also**  
GetKeyboardState

## SetMenu (2.x)

```
BOOL SetMenu(hwnd, hmenu)
HWND hwnd;      /* handle of window */
HMENU hmenu;    /* handle of menu */
```

The SetMenu function sets the given window's menu to the specified menu.

| Parameter | Description                                                                               |
|-----------|-------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window whose menu is to be changed.                                        |
| hmenu     | Identifies the new menu. If this parameter is NULL, the window's current menu is removed. |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The SetMenu function causes the window to be redrawn to reflect the menu change.

SetMenu will not destroy a previous menu. An application should call the DestroyMenu function to accomplish this task.

### Example

```
HMENU hmenu;
```

```
hmenu = LoadMenu(hinst, "My Menu");
SetMenu(hwnd, hmenu);
```

### See Also

DestroyMenu, LoadMenu, LoadMenuIndirect

### SetMenuItemBitmaps (3.0)

```
BOOL SetMenuItemBitmaps(hmenu, idItem, fuFlags, hbmUnchecked, hbmChecked)
HMENU hmenu;          /* handle of menu */
UINT idItem;          /* menu-item identifier */
UINT fuFlags;        /* menu-item flags */
HBITMAP hbmUnchecked; /* handle of unchecked bitmap */
HBITMAP hbmChecked;  /* handle of checked bitmap */
```

The SetMenuItemBitmaps function associates the given bitmaps with a menu item. Whether the menu item is checked or unchecked, Windows displays the appropriate check-mark bitmap next to the menu item.

| Parameter     | Description                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |              |                                                                          |               |                                                                          |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------------|--------------------------------------------------------------------------|---------------|--------------------------------------------------------------------------|
| hmenu         | Identifies the menu.                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |              |                                                                          |               |                                                                          |
| idItem        | Specifies the menu item to be changed, as determined by the fuFlags parameter.                                                                                                                                                                                                                                                                                                                                            |       |         |              |                                                                          |               |                                                                          |
| fuFlags       | Specifies how the idItem parameter is interpreted. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>MF_BYCOMMAND</td><td>The idItem parameter specifies the menu-item identifier (default value).</td></tr><tr><td>MF_BYPOSITION</td><td>The idItem parameter specifies the zero-based position of the menu item.</td></tr></tbody></table> | Value | Meaning | MF_BYCOMMAND | The idItem parameter specifies the menu-item identifier (default value). | MF_BYPOSITION | The idItem parameter specifies the zero-based position of the menu item. |
| Value         | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |              |                                                                          |               |                                                                          |
| MF_BYCOMMAND  | The idItem parameter specifies the menu-item identifier (default value).                                                                                                                                                                                                                                                                                                                                                  |       |         |              |                                                                          |               |                                                                          |
| MF_BYPOSITION | The idItem parameter specifies the zero-based position of the menu item.                                                                                                                                                                                                                                                                                                                                                  |       |         |              |                                                                          |               |                                                                          |
| hbmUnchecked  | Identifies the check-mark bitmap to display when the menu item is not checked.                                                                                                                                                                                                                                                                                                                                            |       |         |              |                                                                          |               |                                                                          |
| hbmChecked    | Identifies the check-mark bitmap to display when the menu item is checked.                                                                                                                                                                                                                                                                                                                                                |       |         |              |                                                                          |               |                                                                          |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

If either the hbmUnchecked or the hbmChecked parameter is NULL, Windows displays nothing next to the menu item for the corresponding attribute. If both parameters are NULL, Windows uses the default check mark when the item is checked and removes the check mark when the item is unchecked.

When the menu is destroyed, these bitmaps are not destroyed; the application must destroy them.

The GetMenuCheckMarkDimensions function retrieves the dimensions of the default check mark used for menu items. The application should use these values to determine the appropriate size for the bitmaps supplied with this function.

#### See Also

GetMenuCheckMarkDimensions

## SetMessageQueue (2.x)

```
BOOL SetMessageQueue(cMsg)
int cMsg; /* size of message queue */
```

The SetMessageQueue function creates a new message queue. It is particularly useful in applications that require a queue that contains more than eight messages (the maximum size of the default queue).

| Parameter | Description                                                                                                      |
|-----------|------------------------------------------------------------------------------------------------------------------|
| cMsg      | Specifies the maximum number of messages that the new queue may contain. This value must not be larger than 120. |

### Returns

The return value is nonzero if the function is successful. If the value specified in the cMsg parameter is larger than 120, the return value is nonzero but the message queue is not created. The return value is zero if an error occurs.

### Comments

The function must be called from an application's WinMain function before any windows are created and before any messages are sent. The SetMessageQueue function destroys the old queue, along with messages it might contain.

If the return value is zero, the application has no queue, because the SetMessageQueue function deletes the original queue before attempting to create a new one. The application must continue calling SetMessageQueue with a smaller queue size until the function returns nonzero.

### See Also

GetMessage, PeekMessage

## SetParent (2.x)

```
HWND SetParent(HWND hWndChild, HWND hWndNewParent)
HWND hWndChild;          /* handle of window whose parent is changing */
HWND hWndNewParent;
/* handle of new parent window
/
```

The SetParent function changes the parent window of the given child window.

| Parameter     | Description                       |
|---------------|-----------------------------------|
| hWndChild     | Identifies the child window.      |
| hWndNewParent | Identifies the new parent window. |

### Returns

The return value is the handle of the previous parent window, if the function is successful.

### Comments

If the window identified by the hWndChild parameter is visible, Windows performs the appropriate redrawing and repainting.

### See Also

[GetParent](#), [IsChild](#)

## SetProp (2.x)

```
BOOL SetProp(HWND hwnd, LPCSTR lpsz, HANDLE hData)
/* handle of window */
/* atom or address of string */
HANDLE hData;
/* handle of data */
```

The SetProp function adds a new entry or changes an existing entry in the property list of the given window. The function adds a new entry to the list if the given character string does not exist already in the list. The new entry contains the string and the handle. Otherwise, the function replaces the string's current handle with the given handle.

| Parameter | Description                                                                                                                                                                                                                                                                                                       |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window whose property list receives the new entry.                                                                                                                                                                                                                                                 |
| lpsz      | Points to a null-terminated string or an atom that identifies a string. If this parameter is an atom, it must be a global atom created by a previous call to the <u>GlobalAddAtom</u> function. The atom, a 16-bit value, must be placed in the low-order <u>word of lpsz</u> ; the high-order word must be zero. |
| hData     | Identifies data to be copied to the property list. The data handle can identify any 16-bit value useful to the application.                                                                                                                                                                                       |

### Returns

The return value is nonzero if the data handle and string are added to the property list. Otherwise, it is zero.

### Comments

Before destroying a window (that is, before processing the WM\_DESTROY message), an application must remove all entries it has added to the property list. The RemoveProp function must be used to remove entries from a property list.

### See Also

GetProp, GlobalAddAtom, RemoveProp

## SetRect (2.x)

```
void SetRect(lprc, nLeft, nTop, nRight, nBottom)
RECT FAR* lprc;          /* address of structure with rectangle to set */
int nLeft;              /* left side */
/
int nTop;               /* top side */
/
int nRight;            /* right side */
/
int nBottom;          /* bottom side */
/
```

The SetRect function sets rectangle coordinates. The action of this function is equivalent to assigning the left, top, right, and bottom arguments to the appropriate members of the RECT structure.

| Parameter | Description                                                                |
|-----------|----------------------------------------------------------------------------|
| lprc      | Points to the <u>RECT</u> structure that contains the rectangle to be set. |
| nLeft     | Specifies the x-coordinate of the upper-left corner.                       |
| nTop      | Specifies the y-coordinate of the upper-left corner.                       |
| nRight    | Specifies the x-coordinate of the lower-right corner.                      |
| nBottom   | Specifies the y-coordinate of the lower-right corner.                      |

### Returns

This function does not return a value.

### Comments

The width of the rectangle, specified by the absolute value of nRight - nLeft, must not exceed 32,767 units. This limit also applies to the height of the rectangle.

### See Also

CopyRect, SetRectEmpty

## SetRectEmpty (2.x)

```
void SetRectEmpty(lprc)
RECT FAR* lprc;          /* address of struct. with rectangle to set to empty */
```

The SetRectEmpty function creates an empty rectangle (all coordinates set to zero).

| Parameter | Description                                                                         |
|-----------|-------------------------------------------------------------------------------------|
| lprc      | Points to the <u>RECT</u> structure that contains the rectangle to be set to empty. |

### Returns

This function does not return a value.

### See Also

CopyRect, SetRect, RECT



## SetScrollPos (2.x)

```
int SetScrollPos(hwnd, fnBar, nPos, fRepaint)
HWND hwnd;          /* handle of window with scroll bar */
int fnBar;          /* scroll bar flag */
int nPos;
/* new position of scroll box
/
BOOL fRepaint;
/* redraw flag
/
```

The SetScrollPos function sets the position of a scroll box (thumb) and, if requested, redraws the scroll bar to reflect the new position of the scroll box.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |        |                                                                                                                           |         |                                                                          |         |                                                                        |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------|---------------------------------------------------------------------------------------------------------------------------|---------|--------------------------------------------------------------------------|---------|------------------------------------------------------------------------|
| hwnd      | Identifies the window whose scroll bar is to be set.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |        |                                                                                                                           |         |                                                                          |         |                                                                        |
| fnBar     | Specifies the scroll bar to be set. This parameter can be one of the following values: <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>SB_CTL</td><td>Sets the position of the scroll box in a scroll bar. In this case, the hwnd parameter must be the handle of a scroll bar.</td></tr><tr><td>SB_HORZ</td><td>Sets the position of the scroll box in a window's horizontal scroll bar.</td></tr><tr><td>SB_VERT</td><td>Sets the position of the scroll box in a window's vertical scroll bar.</td></tr></tbody></table> | Value | Meaning | SB_CTL | Sets the position of the scroll box in a scroll bar. In this case, the hwnd parameter must be the handle of a scroll bar. | SB_HORZ | Sets the position of the scroll box in a window's horizontal scroll bar. | SB_VERT | Sets the position of the scroll box in a window's vertical scroll bar. |
| Value     | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |        |                                                                                                                           |         |                                                                          |         |                                                                        |
| SB_CTL    | Sets the position of the scroll box in a scroll bar. In this case, the hwnd parameter must be the handle of a scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |        |                                                                                                                           |         |                                                                          |         |                                                                        |
| SB_HORZ   | Sets the position of the scroll box in a window's horizontal scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |        |                                                                                                                           |         |                                                                          |         |                                                                        |
| SB_VERT   | Sets the position of the scroll box in a window's vertical scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |        |                                                                                                                           |         |                                                                          |         |                                                                        |
| nPos      | Specifies the new position of the scroll box. It must be within the scrolling range.                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |        |                                                                                                                           |         |                                                                          |         |                                                                        |
| fRepaint  | Specifies whether the scroll bar should be repainted to reflect the new scroll box position. If this parameter is TRUE, the scroll bar is repainted. If it is FALSE, the scroll bar is not repainted.                                                                                                                                                                                                                                                                                                                                                   |       |         |        |                                                                                                                           |         |                                                                          |         |                                                                        |

### Returns

The return value is the previous position of the scroll box, if the function is successful. Otherwise, it is zero.

### Comments

Setting the fRepaint parameter to FALSE is useful whenever the scroll bar will be redrawn by a subsequent call to another function.

### See Also

[GetScrollPos](#), [GetScrollRange](#), [ScrollWindow](#), [SetScrollRange](#)

## SetScrollRange (2.x)

```
void SetScrollRange(HWND hwnd, int fnBar, int nMin, int nMax, BOOL fRedraw)
/* handle of window with scroll bar */
/* scroll bar flag */
int fnBar;
int nMin;
/* minimum scrolling position
/
int nMax;
/* maximum scrolling position
/
BOOL fRedraw;
/* redraw flag
/
```

The SetScrollRange function sets minimum and maximum position values for the given scroll bar. It can also be used to hide or show standard scroll bars.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |        |                                                                                                      |         |                                                     |         |                                                   |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------|------------------------------------------------------------------------------------------------------|---------|-----------------------------------------------------|---------|---------------------------------------------------|
| hwnd      | Identifies a window or a scroll bar, depending on the value of the fnBar parameter.                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |        |                                                                                                      |         |                                                     |         |                                                   |
| fnBar     | Specifies the scroll bar to be set. This parameter can be one of the following values:<br><table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>SB_CTL</td><td>Sets the range of a scroll bar. In this case, the hwnd parameter must be the handle of a scroll bar.</td></tr><tr><td>SB_HORZ</td><td>Sets the range of a window's horizontal scroll bar.</td></tr><tr><td>SB_VERT</td><td>Sets the range of a window's vertical scroll bar.</td></tr></tbody></table> | Value | Meaning | SB_CTL | Sets the range of a scroll bar. In this case, the hwnd parameter must be the handle of a scroll bar. | SB_HORZ | Sets the range of a window's horizontal scroll bar. | SB_VERT | Sets the range of a window's vertical scroll bar. |
| Value     | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |        |                                                                                                      |         |                                                     |         |                                                   |
| SB_CTL    | Sets the range of a scroll bar. In this case, the hwnd parameter must be the handle of a scroll bar.                                                                                                                                                                                                                                                                                                                                                                                        |       |         |        |                                                                                                      |         |                                                     |         |                                                   |
| SB_HORZ   | Sets the range of a window's horizontal scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |        |                                                                                                      |         |                                                     |         |                                                   |
| SB_VERT   | Sets the range of a window's vertical scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |        |                                                                                                      |         |                                                     |         |                                                   |
| nMin      | Specifies the minimum scrolling position.                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |        |                                                                                                      |         |                                                     |         |                                                   |
| nMax      | Specifies the maximum scrolling position.                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |        |                                                                                                      |         |                                                     |         |                                                   |
| fRedraw   | Specifies whether the scroll bar should be redrawn to reflect the change. If this parameter is TRUE, the scroll bar is redrawn. If it is FALSE, the scroll bar is not redrawn.                                                                                                                                                                                                                                                                                                              |       |         |        |                                                                                                      |         |                                                     |         |                                                   |

### Returns

This function does not return a value.

### Comments

An application should not call this function to hide a scroll bar while processing a scroll-bar notification message.

If the call to SetScrollRange immediately follows the call to the SetScrollPos function, the fRedraw parameter in SetScrollPos should be zero, to prevent the scroll bar from being drawn twice.

The default range for a standard scroll bar is 0 through 100. The default range for a scroll bar control is empty (both the nMin and nMax values are zero). The difference between the values specified by the nMin and nMax parameters must not be greater than 32,767.

### See Also

GetScrollPos, GetScrollRange, ScrollWindow, SetScrollPos

## SetSysColors (2.x)

```
void SetSysColors(cDspElements, lpnDspElements, lpdwRgbValues)
int cDspElements; /* number of elements to change */
const int FAR* lpnDspElements;
/* address of array of elements
/
const COLORREF FAR* lpdwRgbValues;
/* address of array of RGB values
/
```

The SetSysColors function sets the system colors for one or more display elements. Display elements are the various parts of a window and the Windows background that appear on the screen.

The SetSysColors function sends a WM\_SYSCOLORCHANGE message to all windows to inform them of the change in color. It also directs Windows to repaint the affected portions of all currently visible windows.

| Parameter      | Description                                                                                                                                                                           |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cDspElements   | Specifies the number of display elements in the array pointed to by the lpnDspElements parameter.                                                                                     |
| lpnDspElements | Points to an array of integers that specify the display elements to be changed. For a list of possible display elements, see the following Comments section.                          |
| lpdwRgbValues  | Points to an array of unsigned long integers that contains the new RGB (red-green-blue) color value for each display element in the array pointed to by the lpnDspElements parameter. |

### Returns

This function does not return a value.

### Comments

The SetSysColors function changes the current Windows session only. The new colors are not saved when Windows terminates.

Following are the display elements that may be used in the lpnDspElements array:

| Value                            | Meaning                                                                                                     |
|----------------------------------|-------------------------------------------------------------------------------------------------------------|
| <u>COLOR_ACTIVEBORDER</u>        | Active window border.                                                                                       |
| <u>COLOR_ACTIVECAPTION</u>       | Active window title.                                                                                        |
| <u>COLOR_APPWORKSPACE</u>        | Background color of multiple document interface (MDI) applications.                                         |
| <u>COLOR_BACKGROUND</u>          | Desktop.                                                                                                    |
| <u>COLOR_BTNFACE</u>             | Face shading on push buttons.                                                                               |
| <u>COLOR_BTNHIGHLIGHT</u>        | Selected button in a control.                                                                               |
| <u>COLOR_BTNSHADOW</u>           | Edge shading on push buttons.                                                                               |
| <u>COLOR_BTNTEXT</u>             | Text on push buttons.                                                                                       |
| <u>COLOR_CAPTIONTEXT</u>         | Text in title bar, size button, scroll-bar arrow button.                                                    |
| <u>COLOR_GRAYTEXT</u>            | Grayed (dimmed) text. This color is zero if the current display driver does not support a solid gray color. |
| <u>COLOR_HIGHLIGHT</u>           | Background of selected item in a control.                                                                   |
| <u>COLOR_HIGHLIGHTTEXT</u>       | Text of selected item in a control.                                                                         |
| <u>COLOR_INACTIVEBORDER</u>      | Inactive window border.                                                                                     |
| <u>COLOR_INACTIVECAPTION</u>     | Inactive window title.                                                                                      |
| <u>COLOR_INACTIVECAPTIONTEXT</u> | Color of text in an inactive title.                                                                         |
| <u>COLOR_MENU</u>                | Menu background.                                                                                            |
| <u>COLOR_MENUTEXT</u>            | Text in menus.                                                                                              |
| <u>COLOR_SCROLLBAR</u>           | Scroll-bar gray area.                                                                                       |
| <u>COLOR_WINDOW</u>              | Window background.                                                                                          |
| <u>COLOR_WINDOWFRAME</u>         | Window frame.                                                                                               |
| <u>COLOR_WINDOWTEXT</u>          | Text in windows.                                                                                            |

### Example

The following example changes the window background to black and the text in the window to green:

```
int aiDspElements[2];
DWORD aRgbValues[2];

aiDspElements[0] = COLOR_WINDOW;
aRgbValues[0] = RGB(
    0x00, /* red */
    0x00, /* green */
    0x00); /* blue */
aiDspElements[1] = COLOR_WINDOWTEXT;
aRgbValues[1] = RGB(
    0x00, /* red */
    0xff, /* green */
    0x00); /* blue */
SetSysColors(2, aiDspElements, aRgbValues);
```

See Also

[GetSysColor](#), [WM\\_SYSCOLORCHANGE](#)

Windows 3.1 changes

The following constants have been added:

Value

COLOR\_BTNHIGHLIGHT

COLOR\_INACTIVECAPTIONTEXT

Meaning

Selected button in a control.

Color of text in an inactive caption.

COLOR\_ACTIVEBORDER 10

Active window border.

COLOR\_ACTIVEBORDER 10

COLOR\_ACTIVECAPTION 2

Active window title.



COLOR\_ACTIVECAPTION 2

COLOR\_APPWORKSPACE 12

Background color of multiple document interface (MDI) applications.

COLOR\_APPWORKSPACE 12

COLOR\_BACKGROUND 1  
Desktop.

COLOR\_BACKGROUND 1

COLOR\_BTNFACE 15

Face shading on push buttons.

COLOR\_BTNFACE 15

COLOR\_BTNHIGHLIGHT 20

Selected button in a control.



COLOR\_BTNHIGHLIGHT 20

COLOR\_BTNSHADOW 16

Edge shading on push buttons.

COLOR\_BTNSHADOW 16

COLOR\_BTNTEXT 18

Text on push buttons.

COLOR\_BTNTEXT 18

COLOR\_CAPTIONTEXT 9

Text in title bar, size button, scroll-bar arrow button.

COLOR\_CAPTIONTEXT 9

COLOR\_GRAYTEXT 17

Grayed (dimmed) text. This color is zero if the current display driver does not support a solid gray color.



COLOR\_GRAYTEXT 17

COLOR\_HIGHLIGHT 13

Background of selected item in a control.

COLOR\_HIGHLIGHT 13

COLOR\_HIGHLIGHTTEXT 14

Text of selected item in a control.

COLOR\_HIGHLIGHTTEXT 14

COLOR\_INACTIVEBORDER 11

Inactive window border.

COLOR\_INACTIVEBORDER 11

COLOR\_INACTIVECAPTION 3

Inactive window title.



COLOR\_INACTIVECAPTION 3

COLOR\_INACTIVECAPTIONTEXT 19

Color of text in an inactive title.

COLOR\_INACTIVECAPTIONTEXT 19

COLOR\_MENU 4  
Menu background.

COLOR\_MENU 4

COLOR\_MENUTEXT 7

Text in menus.

COLOR\_MENUTEXT 7

COLOR\_SCROLLBAR 0  
Scroll-bar gray area.



COLOR\_SCROLLBAR 0

COLOR\_WINDOW 5  
Window background.

COLOR\_WINDOW 5

COLOR\_WINDOWFRAME 6

Window frame.

COLOR\_WINDOWFRAME 6

COLOR\_WINDOWTEXT 8

Text in windows.

COLOR\_WINDOWTEXT 8

## SetSysModalWindow (2.x)

```
HWND SetSysModalWindow(HWND)
HWND hwnd; /* handle of window to become system modal */
```

The SetSysModalWindow function makes the given window the system-modal window.

| Parameter | Description                                    |
|-----------|------------------------------------------------|
| hwnd      | Identifies the window to be made system modal. |

### Returns

The return value is the handle of the window that was previously the system-modal window, if the function is successful.

### Comments

If another window is made the active window (for example, the system-modal window creates a dialog box that becomes the active window), the active window becomes the system-modal window. When the original window becomes active again, it is once again the system-modal window. To end the system-modal state, destroy the system-modal window.

If a WH\_JOURNALRECORD hook is in place when SetSysModalWindow is called, the hook is called with a hook code of HC\_SYSMODALON (for turning on the system-modal window) or HC\_SYSMODALOFF (for turning off the system-modal window).

### See Also

GetSysModalWindow



## SetTimer (2.x)

```
UINT SetTimer(hwnd, idTimer, uTimeout, tmprc)
HWND hwnd;          /* handle of window for timer messages */
UINT idTimer;       /* timer
identifier
/
UINT uTimeout;
* time-out duration
/
TIMERPROC tmprc;
* instance address of timer procedure
/
```

The SetTimer function installs a system timer. A time-out value is specified, and every time a time-out occurs, the system posts a WM\_TIMER message to the installing application's message queue or passes the message to an application-defined TimerProc callback function.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window to be associated with the timer. If the tmprc parameter is NULL, the window procedure associated with this window receives the <u>WM_TIMER</u> messages generated by the timer. If this parameter is NULL, no window is associated with the timer.                                                                                                                                |
| idTimer   | Specifies a nonzero timer identifier. If the hwnd parameter is NULL, this parameter is ignored.                                                                                                                                                                                                                                                                                                         |
| uTimeout  | Specifies the time-out value, in milliseconds.                                                                                                                                                                                                                                                                                                                                                          |
| tmprc     | Specifies the procedure-instance address of the callback function that processes the <u>WM_TIMER</u> messages. If this parameter is NULL, the <u>WM_TIMER</u> messages are placed in the application's message queue and the hwnd member of the <u>MSG</u> structure contains the window handle specified in hwnd. For more information, see the description of the <u>TimerProc</u> callback function. |

### Returns

The return value is the identifier of the new timer if hwnd is NULL and the function is successful. An application passes this value to the KillTimer function to kill the timer. The return value is nonzero if hwnd is a valid window handle and the function is successful. Otherwise, the return value is zero.

### Comments

Timers are a limited global resource; therefore, it is important that an application check the value returned by the SetTimer function to verify that a timer is available.

The tmprc parameter must specify a procedure-instance address of the callback function, and the callback function must be exported in the application's module-definition file. A procedure-instance address can be created by using the MakeProcInstance function. The callback function must use the Pascal calling convention and must be declared as FAR.

### Example

The following example installs a system timer. The system will pass WM\_TIMER messages generated by the timer to the "MyTimerProc" callback function.

```
TIMERPROC lpfnMyTimerProc;

lpfnMyTimerProc = (TIMERPROC) MakeProcInstance(MyTimerProc, hinst);
SetTimer(hwnd, ID_MYTIMER, 5000, lpfnMyTimerProc);
```

### See Also

KillTimer, MakeProcInstance, TimerProc, MSG, WM\_TIMER

## SetWindowLong (2.x)

LONG SetWindowLong(hwnd, nOffset, nVal)

HWND hwnd; /\* handle of window \*/

int nOffset;

/\* offset of value to set \*/

LONG nVal;

\* new value

/

\*

The SetWindowLong function places a long value at the specified offset into the extra window memory of the given window. Extra window memory is reserved by specifying a nonzero value in the cbWndExtra member of the WNDCLASS structure used with the RegisterClass function.

| Parameter     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |             |                       |           |              |             |                                      |       |         |             |                                                    |               |                                                                                |          |                                                                                              |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|-------------|-----------------------|-----------|--------------|-------------|--------------------------------------|-------|---------|-------------|----------------------------------------------------|---------------|--------------------------------------------------------------------------------|----------|----------------------------------------------------------------------------------------------|
| hwnd          | Identifies the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |             |                       |           |              |             |                                      |       |         |             |                                                    |               |                                                                                |          |                                                                                              |
| nOffset       | Specifies the zero-based byte offset of the value to change. Valid values are in the range zero through the number of bytes of extra window memory, minus four (for example, if 12 or more bytes of extra memory were specified, a value of 8 would be an index to the third long integer), or one of the following values:<br><table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>GWL_EXSTYLE</td><td>Extended window style</td></tr><tr><td>GWL_STYLE</td><td>Window style</td></tr><tr><td>GWL_WNDPROC</td><td>Long pointer to the window procedure</td></tr></tbody></table> The following values are also available when the hwnd parameter identifies a dialog box:<br><table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>DWL_DLGPROC</td><td>Specifies the address of the dialog box procedure.</td></tr><tr><td>DWL_MSGRESULT</td><td>Specifies the return value of a message processed in the dialog box procedure.</td></tr><tr><td>DWL_USER</td><td>Specifies extra information that is private to the application, such as handles or pointers.</td></tr></tbody></table> | Value | Meaning | GWL_EXSTYLE | Extended window style | GWL_STYLE | Window style | GWL_WNDPROC | Long pointer to the window procedure | Value | Meaning | DWL_DLGPROC | Specifies the address of the dialog box procedure. | DWL_MSGRESULT | Specifies the return value of a message processed in the dialog box procedure. | DWL_USER | Specifies extra information that is private to the application, such as handles or pointers. |
| Value         | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |             |                       |           |              |             |                                      |       |         |             |                                                    |               |                                                                                |          |                                                                                              |
| GWL_EXSTYLE   | Extended window style                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |             |                       |           |              |             |                                      |       |         |             |                                                    |               |                                                                                |          |                                                                                              |
| GWL_STYLE     | Window style                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |             |                       |           |              |             |                                      |       |         |             |                                                    |               |                                                                                |          |                                                                                              |
| GWL_WNDPROC   | Long pointer to the window procedure                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |             |                       |           |              |             |                                      |       |         |             |                                                    |               |                                                                                |          |                                                                                              |
| Value         | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |             |                       |           |              |             |                                      |       |         |             |                                                    |               |                                                                                |          |                                                                                              |
| DWL_DLGPROC   | Specifies the address of the dialog box procedure.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |             |                       |           |              |             |                                      |       |         |             |                                                    |               |                                                                                |          |                                                                                              |
| DWL_MSGRESULT | Specifies the return value of a message processed in the dialog box procedure.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |             |                       |           |              |             |                                      |       |         |             |                                                    |               |                                                                                |          |                                                                                              |
| DWL_USER      | Specifies extra information that is private to the application, such as handles or pointers.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |             |                       |           |              |             |                                      |       |         |             |                                                    |               |                                                                                |          |                                                                                              |
| nVal          | Specifies the long value to place in the window's reserved memory.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |             |                       |           |              |             |                                      |       |         |             |                                                    |               |                                                                                |          |                                                                                              |

### Returns

The return value is the previous value of the specified long integer, if the function is successful. Otherwise, it is zero.

### Comments

If the SetWindowLong function and the GWL\_WNDPROC index are used to set a new window procedure, that procedure must have the window-procedure form and be exported in the module-definition file of the application. For more information, see the description of the RegisterClass function.

Calling SetWindowLong with the GCL\_WNDPROC index creates a subclass of the window class used to create the window. An application should not attempt to create a window subclass for standard Windows controls such as combo boxes and buttons.

An application should not use this function to set the WS\_DISABLE style for a window. Instead, the application should use the EnableWindow function.

To access any extra 4-byte values allocated when the window-class structure was created, use a positive byte offset as the index specified by the nOffset parameter, starting at 0 for the first 4-byte value in the extra space, 4 for the next 4-byte value, and so on.

An application can use the DWL\_MSGRESULT value to return values from a dialog box procedure's window procedure. Typically, a dialog box procedure must return TRUE in order for a value to be returned to the sender of the message. Some messages, however, return a value in the Boolean return value of the dialog box procedure. The following messages return values in the return value of the dialog box procedure:

WM\_CHARTOITEM  
WM\_COMPAREITEM  
WM\_CTLCOLOR  
WM\_INITDIALOG

WM\_QUERYDRAGICON  
WM\_VKEYTOITEM

Example

The following example shows how to use the SetWindowLong function with the DWL\_MSGRESULT value to return a value from a dialog box procedure. Applications often include a switch statement to handle the messages that return values in the Boolean return value of the dialog box procedure, even when the dialog box procedure does not process these messages. This practice makes it easy to revise the dialog box procedure to handle the message and has a negligible effect on speed and memory.

```
BOOL CALLBACK MyDlgProc(hwndDlg, msg, wParam, lParam)
HWND hwndDlg;
UINT msg;
WPARAM wParam;
LPARAM lParam;
{
    BOOL fProcessed = FALSE;
    LRESULT lResult;

    /*
    * To return a value for a specific message, set lResult to the
    * return value and fProcessed to TRUE.
    */

    switch (msg) {
    . /* process messages */
    .
    case WM_QUERYENDSESSION:

    /*
    * Example: Do not allow the system to terminate
    * while the dialog box is displayed.
    */

    fProcessed = TRUE;
    lResult = (LRESULT) (UINT) FALSE;
    break;

    default:
    break;
    }

    if (fProcessed) {
    switch (msg) {
    case WM_CTLCOLOR:
    case WM_COMPAREITEM:
    case WM_VKEYTOITEM:
    case WM_CHARTOITEM:
    case WM_QUERYDRAGICON:
    case WM_INITDIALOG:
    return (BOOL) LOWORD(lResult);
    default:
    SetWindowLong(hwndDlg, DWL_MSGRESULT, (LPARAM) lResult);
    }
    }
    return fProcessed;
}
```

See Also

EnableWindow, GetWindowLong, RegisterClass, SetWindowWord

### SetWindowPlacement (3.1)

```
BOOL SetWindowPlacement(HWND hwnd, lpwndpl)           /* handle of the window */
HWND hwnd;
const WINDOWPLACEMENT FAR* lpwndpl;
/* address of position data
/
```

The SetWindowPlacement function sets the show state and the normal (restored), minimized, and maximized positions for a window.

| Parameter | Description                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window.                                                                        |
| lpwndpl   | Points to a <u>WINDOWPLACEMENT</u> structure that specifies the new show state and positions. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### See Also

GetWindowPlacement, WINDOWPLACEMENT

■ **SetWindowPos (2.x)**

```

BOOL SetWindowPos(HWND hwnd, HWND hwndInsertAfter, int x, int y, int cx, int cy, DWORD fuFlags)
/* handle of window */
/*
placement-order handle
/
int x;
* horizontal position
/
int y;
* vertical position
/
int cx;
* width
/
int cy;
* height
/
UINT fuFlags;
* window-positioning flags
/

```

The SetWindowPos function changes the size, position, and Z-order of child, pop-up, and top-level windows. These windows are ordered according to their appearance on the screen; the window on top receives the highest rank and is the first window in the Z-order.

| Parameter       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|----------------------------------------------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd            | Identifies the window to be positioned.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
| hwndInsertAfter | Identifies the window to precede the positioned window in the Z-order. This parameter must be a window handle or one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
|                 | <table border="0"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>HWND_BOTTOM</td> <td>Places the window at the bottom of the Z-order. If hwnd identifies a topmost window, the window loses its topmost status; the system places the window at the bottom of all other windows.</td> </tr> <tr> <td>HWND_TOP</td> <td>Places the window at the top of the Z-order.</td> </tr> <tr> <td>HWND_TOPMOST</td> <td>Places the window above all non-topmost windows. The window maintains its topmost position even when it is deactivated.</td> </tr> <tr> <td>HWND_NOTOPMOST</td> <td>Repositions the window to the top of all non-topmost windows (that is, behind all topmost windows). This flag has no effect if the window is already a non-topmost window.</td> </tr> </tbody> </table> | Value | Meaning | HWND_BOTTOM   | Places the window at the bottom of the Z-order. If hwnd identifies a topmost window, the window loses its topmost status; the system places the window at the bottom of all other windows. | HWND_TOP       | Places the window at the top of the Z-order. | HWND_TOPMOST   | Places the window above all non-topmost windows. The window maintains its topmost position even when it is deactivated.                        | HWND_NOTOPMOST | Repositions the window to the top of all non-topmost windows (that is, behind all topmost windows). This flag has no effect if the window is already a non-topmost window. |
| Value           | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
| HWND_BOTTOM     | Places the window at the bottom of the Z-order. If hwnd identifies a topmost window, the window loses its topmost status; the system places the window at the bottom of all other windows.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
| HWND_TOP        | Places the window at the top of the Z-order.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
| HWND_TOPMOST    | Places the window above all non-topmost windows. The window maintains its topmost position even when it is deactivated.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
| HWND_NOTOPMOST  | Repositions the window to the top of all non-topmost windows (that is, behind all topmost windows). This flag has no effect if the window is already a non-topmost window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
|                 | See the following Comments section for rules about how this parameter is used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
| x               | Specifies the new position of the left side of the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
| y               | Specifies the new position of the top of the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
| cx              | Specifies the new width of the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
| cy              | Specifies the new height of the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
| fuFlags         | Specifies the window sizing and positioning options. This parameter can be a combination of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
|                 | <table border="0"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>SWP_DRAWFRAME</td> <td>Draws a frame (defined in the window's class description) around the window.</td> </tr> <tr> <td>SWP_HIDEWINDOW</td> <td>Hides the window.</td> </tr> <tr> <td>SWP_NOACTIVATE</td> <td>Does not activate the window. If this flag is not set, the window is activated and moved to the top of either the topmost or non-topmost group</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                | Value | Meaning | SWP_DRAWFRAME | Draws a frame (defined in the window's class description) around the window.                                                                                                               | SWP_HIDEWINDOW | Hides the window.                            | SWP_NOACTIVATE | Does not activate the window. If this flag is not set, the window is activated and moved to the top of either the topmost or non-topmost group |                |                                                                                                                                                                            |
| Value           | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
| SWP_DRAWFRAME   | Draws a frame (defined in the window's class description) around the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
| SWP_HIDEWINDOW  | Hides the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |
| SWP_NOACTIVATE  | Does not activate the window. If this flag is not set, the window is activated and moved to the top of either the topmost or non-topmost group                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |               |                                                                                                                                                                                            |                |                                              |                |                                                                                                                                                |                |                                                                                                                                                                            |

|                             |                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             | (depending on the setting of the <code>hwndInsertAfter</code> parameter).                                                                                                                                                                                                                                                                                                                            |
| <code>SWP_NOMOVE</code>     | Retains the current position (ignores the <code>x</code> and <code>y</code> parameters).                                                                                                                                                                                                                                                                                                             |
| <code>SWP_NOSIZE</code>     | Retains the current size (ignores the <code>cx</code> and <code>cy</code> parameters).                                                                                                                                                                                                                                                                                                               |
| <code>SWP_NOREDRA</code>    | Does not redraw changes. If this flag is set, no repainting of any kind occurs. This applies to the client area, the non-client area (including the title and scroll bars), and any part of the parent window uncovered as a result of the moved window. When this flag is set, the application must explicitly invalidate or redraw any parts of the window and parent window that must be redrawn. |
| <code>SWP_NOZORDER</code>   | Retains the current ordering (ignores the <code>hwndInsertAfter</code> parameter).                                                                                                                                                                                                                                                                                                                   |
| <code>SWP_SHOWWINDOW</code> | Displays the window.                                                                                                                                                                                                                                                                                                                                                                                 |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

If the `SWP_SHOWWINDOW` or the `SWP_HIDEWINDOW` flags are set, the window cannot be moved or sized.

All coordinates for child windows are client coordinates (relative to the upper-left corner of the parent window's client area).

A window can be made a topmost window either by setting the `hwndInsertAfter` parameter to `HWND_TOPMOST` and ensuring that the `SWP_NOZORDER` flag is not set, or by setting a window's Z-order so that it is above any existing topmost windows. When a non-topmost window is made topmost, its owned windows are also made topmost. Its owners are not changed.

If neither `SWP_NOACTIVATE` nor `SWP_NOZORDER` is specified (that is, when the application requests that a window be simultaneously activated and placed in the specified Z-order), the value specified in `hwndInsertAfter` is used only in the following circumstances:

- Neither `HWND_TOPMOST` or `HWND_NOTOPMOST` is specified in the `hwndInsertAfter` parameter.
- The window specified in the `hwnd` parameter is not the active window.

An application cannot activate an inactive window without also bringing it to the top of the Z-order. Applications can change the Z-order of an activated window without restrictions or activate a window and then move it to the top of the topmost or non-topmost windows.

A topmost window is no longer topmost if it is repositioned to the bottom (`HWND_BOTTOM`) of the Z-order or after any non-topmost window. When a topmost window is made non-topmost, all of its owners and its owned windows are also made non-topmost windows.

A non-topmost window may own a topmost window, but not vice versa. Any window (for example, a dialog box) owned by a topmost window is itself made a topmost window, to ensure that all owned windows stay above their owner.

#### Example

The following example sets the size of a window equal to one-fourth the size of the desktop and then positions the window in the upper-left corner of the desktop:

```
RECT rect;
```

```
GetWindowRect(GetDesktopWindow(), &rect);
SetWindowPos(hwnd, (HWND) NULL, 0, 0,
    rect.right / 2, rect.bottom / 2,
    SWP_NOZORDER | SWP_NOACTIVATE);
```

#### See Also

[BringWindowToTop](#), [GetWindowRect](#), [MoveWindow](#)

### Windows 3.1 changes

If the `hwndInsertAfter` parameter is `HWND_TOPMOST`, the system places the window identified by the `hwnd` parameter above all non-topmost windows. The window maintains its topmost position even when the window is deactivated. If the `hwndInsertAfter` parameter is `HWND_BOTTOM` and `hwnd` identifies a topmost window, the window loses its topmost status--the system places the window at the bottom of all other windows.

The following window-positioning flags are new for Windows version 3.1:

| Value                       | Meaning                                                                                                                                                                                                 |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>HWND_BOTTOM</code>    | Places the window at the bottom of the Z-order. If <code>hwnd</code> identifies a topmost window, the window loses its topmost status--the system places the window at the bottom of all other windows. |
| <code>HWND_TOP</code>       | Places the window at the top of the Z-order.                                                                                                                                                            |
| <code>HWND_TOPMOST</code>   | Places the window above all non-topmost windows. The window maintains its topmost position even when the window is deactivated.                                                                         |
| <code>HWND_NOTOPMOST</code> | Repositions the window to the top of all non-topmost windows (that is, behind all topmost window).                                                                                                      |

## HWND\_BOTTOM

Places the window at the bottom of the Z-order. If hwnd identifies a topmost window, the window loses its topmost status; the system places the window at the bottom of all other windows.



HWND\_TOP

Places the window at the top of the Z-order.

## HWND\_TOPMOST

Places the window above all non-topmost windows. The window maintains its topmost position even when it is deactivated.

## HWND\_NOTOPMOST

Repositions the window to the top of all non-topmost windows (that is, behind all topmost windows). This flag has no effect if the window is already a non-topmost window.

## SetWindowsHook (2.x)

HHOOK SetWindowsHook(idHook, hkprc)

int idHook; /\* type of hook to install \*/

HOOKPROC hkprc;

\* filter function procedure-instance address

/

The SetWindowsHook function is obsolete but has been retained for backward compatibility with Windows versions 3.0 and earlier. Applications written for Windows version 3.1 should use the SetWindowsHookEx function.

The SetWindowsHook function installs an application-defined hook function into a hook chain.

| Parameter                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|-----------------------|----------------------------------------------------------------------------------------------------------------------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------|-----------------|-------------------------------------------------------------------------------------------------------------------|----------------------|------------------------------------------------------------------------------------------------------------------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------|--------------------|---------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------------------------------------------------------------------------------------------------------|---------------------|-------------------------------------------------------------------------------------------------------------------|-----------------|---------------------------------------------------------------------------------------------------------------------------|------------------------|------------------------------------------------------------------------------------------------------------------------------|
| idHook                    | Specifies the type of hook to be installed. This parameter can be one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
|                           | <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td><u>WH_CALLWNDPROC</u></td> <td>Installs a window-procedure filter. For more information, see the description of the <u>CallWndProc</u> callback function.</td> </tr> <tr> <td><u>WH_CBT</u></td> <td>Installs a computer-based training (CBT) filter. For more information, see the description of the <u>CBTProc</u> callback function.</td> </tr> <tr> <td><u>WH_DEBUG</u></td> <td>Installs a debugging filter. For more information, see the description of the <u>DebugProc</u> callback function.</td> </tr> <tr> <td><u>WH_GETMESSAGE</u></td> <td>Installs a message filter. For more information, see the description of the <u>GetMsgProc</u> callback function.</td> </tr> <tr> <td><u>WH_HARDWARE</u></td> <td>Installs a nonstandard hardware-message filter. For more information, see the description of the <u>HardwareProc</u> callback function.</td> </tr> <tr> <td><u>WH_JOURNALPLAYBACK</u></td> <td>Installs a journaling playback filter. For more information, see the description of the <u>JournalPlaybackProc</u> callback function.</td> </tr> <tr> <td><u>WH_JOURNALRECORD</u></td> <td>Installs a journaling record filter. For more information, see the description of the <u>JournalRecordProc</u> callback function.</td> </tr> <tr> <td><u>WH_KEYBOARD</u></td> <td>Installs a keyboard filter. For more information, see the description of the <u>KeyboardProc</u> callback function.</td> </tr> <tr> <td><u>WH_MOUSE</u></td> <td>Installs a mouse-message filter. For more information, see the description of the <u>MouseProc</u> callback function.</td> </tr> <tr> <td><u>WH_MSGFILTER</u></td> <td>Installs a message filter. For more information, see the description of the <u>MessageProc</u> callback function.</td> </tr> <tr> <td><u>WH_SHELL</u></td> <td>Installs a shell-application filter. For more information, see the description of the <u>ShellProc</u> callback function.</td> </tr> <tr> <td><u>WH_SYSMSGFILTER</u></td> <td>Installs a system-wide message filter. For more information, see the description of the <u>SysMsgProc</u> callback function.</td> </tr> </tbody> </table> | Value | Meaning | <u>WH_CALLWNDPROC</u> | Installs a window-procedure filter. For more information, see the description of the <u>CallWndProc</u> callback function. | <u>WH_CBT</u> | Installs a computer-based training (CBT) filter. For more information, see the description of the <u>CBTProc</u> callback function. | <u>WH_DEBUG</u> | Installs a debugging filter. For more information, see the description of the <u>DebugProc</u> callback function. | <u>WH_GETMESSAGE</u> | Installs a message filter. For more information, see the description of the <u>GetMsgProc</u> callback function. | <u>WH_HARDWARE</u> | Installs a nonstandard hardware-message filter. For more information, see the description of the <u>HardwareProc</u> callback function. | <u>WH_JOURNALPLAYBACK</u> | Installs a journaling playback filter. For more information, see the description of the <u>JournalPlaybackProc</u> callback function. | <u>WH_JOURNALRECORD</u> | Installs a journaling record filter. For more information, see the description of the <u>JournalRecordProc</u> callback function. | <u>WH_KEYBOARD</u> | Installs a keyboard filter. For more information, see the description of the <u>KeyboardProc</u> callback function. | <u>WH_MOUSE</u> | Installs a mouse-message filter. For more information, see the description of the <u>MouseProc</u> callback function. | <u>WH_MSGFILTER</u> | Installs a message filter. For more information, see the description of the <u>MessageProc</u> callback function. | <u>WH_SHELL</u> | Installs a shell-application filter. For more information, see the description of the <u>ShellProc</u> callback function. | <u>WH_SYSMSGFILTER</u> | Installs a system-wide message filter. For more information, see the description of the <u>SysMsgProc</u> callback function. |
| Value                     | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
| <u>WH_CALLWNDPROC</u>     | Installs a window-procedure filter. For more information, see the description of the <u>CallWndProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
| <u>WH_CBT</u>             | Installs a computer-based training (CBT) filter. For more information, see the description of the <u>CBTProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
| <u>WH_DEBUG</u>           | Installs a debugging filter. For more information, see the description of the <u>DebugProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
| <u>WH_GETMESSAGE</u>      | Installs a message filter. For more information, see the description of the <u>GetMsgProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
| <u>WH_HARDWARE</u>        | Installs a nonstandard hardware-message filter. For more information, see the description of the <u>HardwareProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
| <u>WH_JOURNALPLAYBACK</u> | Installs a journaling playback filter. For more information, see the description of the <u>JournalPlaybackProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
| <u>WH_JOURNALRECORD</u>   | Installs a journaling record filter. For more information, see the description of the <u>JournalRecordProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
| <u>WH_KEYBOARD</u>        | Installs a keyboard filter. For more information, see the description of the <u>KeyboardProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
| <u>WH_MOUSE</u>           | Installs a mouse-message filter. For more information, see the description of the <u>MouseProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
| <u>WH_MSGFILTER</u>       | Installs a message filter. For more information, see the description of the <u>MessageProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
| <u>WH_SHELL</u>           | Installs a shell-application filter. For more information, see the description of the <u>ShellProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
| <u>WH_SYSMSGFILTER</u>    | Installs a system-wide message filter. For more information, see the description of the <u>SysMsgProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |
| hkprc                     | Specifies the procedure-instance address of the application-defined hook procedure to be installed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                       |                                                                                                                            |               |                                                                                                                                     |                 |                                                                                                                   |                      |                                                                                                                  |                    |                                                                                                                                         |                           |                                                                                                                                       |                         |                                                                                                                                   |                    |                                                                                                                     |                 |                                                                                                                       |                     |                                                                                                                   |                 |                                                                                                                           |                        |                                                                                                                              |

### Returns

The return value is a handle of the installed hook, if the function is successful. Otherwise, it is NULL.

### Comments

Before terminating, an application must call the UnhookWindowsHook function to free system resources associated with the hook.

The WH\_CALLWNDPROC hook affects system performance. It is supplied for debugging purposes only.

The system hooks are a shared resource. Installing a hook affects all applications. Most hook functions must be in libraries. The only exception is WH\_MSGFILTER, which is task-specific. System hooks should be restricted to special-purpose applications or to use as a development aid during debugging of an application. Libraries that no longer need the hook should remove the filter function.

To install a filter function, the SetWindowsHook function must receive a procedure-instance address of the function and the function must be exported in the library's module-definition file. A task must use the MakeProcInstance function to get a procedure-instance address. A dynamic-link library can pass the procedure address directly.

See Also

DefHookProc, GetProcAddress, MakeProcInstance, MessageBox, PeekMessage, PostMessage, SendMessage, SetWindowsHookEx, UnhookWindowsHook

## Windows 3.1 changes

This function returns an HHOOK value for Windows 3.1. Prior to Windows 3.1 it returned a HOOKPROC value.

The following new hook types have been added:

| Value       | Meaning                                                                                                                                  |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------|
| WH_CBT      | Installs a Computer-Based Training (CBT) filter. For more information, see the description of the <u>CBTProc</u> callback function.      |
| WH_DEBUG    | Installs a debugging filter. For more information, see the description of the <u>DebugProc</u> callback function.                        |
| WH_HARDWARE | Installs a non-standard hardware-message filter. For more information, see the description of the <u>HardwareProc</u> callback function. |
| WH_MOUSE    | Installs a mouse-message filter. For more information, see the description of the <u>MouseProc</u> callback function.                    |
| WH_SHELL    | Installs a shell-application filter. For more information, see the description of the <u>ShellProc</u> callback function.                |

#### WH\_CALLWNDPROC 4

Installs a window-procedure filter. For more information, see the description of the [CallWndProc](#) callback function.

WH\_CALLWNDPROC 4



WH\_CBT 5

Installs a computer-based training (CBT) filter. For more information, see the description of the CBTProc callback function.

WH\_CBT 5

WH\_DEBUG 9

Installs a debugging filter. For more information, see the description of the [DebugProc](#) callback function.

WH\_DEBUG 9

WH\_GETMESSAGE 3

Installs a message filter. For more information, see the description of the [GetMsgProc](#) callback function.

WH\_GETMESSAGE 3

WH\_HARDWARE 8

Installs a nonstandard hardware-message filter. For more information, see the description of the HardwareProc callback function.

WH\_HARDWARE 8



## WH\_JOURNALPLAYBACK 1

Installs a journaling playback filter. For more information, see the description of the [JournalPlaybackProc](#) callback function.

WH\_JOURNALPLAYBACK 1

WH\_JOURNALRECORD 0

Installs a journaling record filter. For more information, see the description of the JournalRecordProc callback function.

WH\_JOURNALRECORD 0

WH\_KEYBOARD 2

Installs a keyboard filter. For more information, see the description of the KeyboardProc callback function.

WH\_KEYBOARD 2

WH\_MOUSE 7

Installs a mouse-message filter. For more information, see the description of the MouseProc callback function.

WH\_MOUSE 7



WH\_MSGFILTER (-1)

Installs a message filter. For more information, see the description of the MessageProc callback function.

WH\_MSGFILTER (-1)

WH\_SHELL 10

Installs a shell-application filter. For more information, see the description of the ShellProc callback function.

WH\_SHELL 10

## WH\_SYSMMSGFILTER 6

Installs a system-wide message filter. For more information, see the description of the [SysMsgProc](#) callback function.

WH\_SYSMMSGFILTER 6

### SetWindowsHookEx (3.1)

```

HHOOK SetWindowsHookEx(idHook, hkprc, hinst, htask)
int idHook; /* type of hook to install */
HOOKPROC hkprc; /* procedure-instance address of filter function */
HINSTANCE hinst; /* handle of application instance */
HTASK htask; /* task to install the hook for */

```

The SetWindowsHookEx function installs an application-defined hook function into a hook chain. This function is an extended version of the [SetWindowsHook](#) function.

| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|----------------|-------------------------------------------------------------------------------------------------------------------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------|----------|----------------------------------------------------------------------------------------------------------------------------|---------------|---------------------------------------------------------------------------------------------------------------------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------|------------------|--------------------------------------------------------------------------------------------------------------------------------------------|-------------|------------------------------------------------------------------------------------------------------------------------------|----------|--------------------------------------------------------------------------------------------------------------------------------|--------------|----------------------------------------------------------------------------------------------------------------------------|----------|------------------------------------------------------------------------------------------------------------------------------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------|
| idHook             | Specifies the type of hook to be installed. This parameter can be one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
|                    | <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>WH_CALLWNDPROC</td> <td>Installs a window-procedure filter. For more information, see the description of the <a href="#">CallWndProc</a> callback function.</td> </tr> <tr> <td>WH_CBT</td> <td>Installs a computer-based training (CBT) filter. For more information, see the description of the <a href="#">CBTProc</a> callback function.</td> </tr> <tr> <td>WH_DEBUG</td> <td>Installs a debugging filter. For more information, see the description of the <a href="#">DebugProc</a> callback function.</td> </tr> <tr> <td>WH_GETMESSAGE</td> <td>Installs a message filter. For more information, see the description of the <a href="#">GetMsgProc</a> callback function.</td> </tr> <tr> <td>WH_HARDWARE</td> <td>Installs a nonstandard hardware-message filter. For more information, see the description of the <a href="#">HardwareProc</a> callback function.</td> </tr> <tr> <td>WH_JOURNALPLAYBACK</td> <td>Installs a journaling playback filter. For more information, see the description of the <a href="#">JournalPlaybackProc</a> callback function.</td> </tr> <tr> <td>WH_JOURNALRECORD</td> <td>Installs a journaling record filter. For more information, see the description of the <a href="#">JournalRecordProc</a> callback function.</td> </tr> <tr> <td>WH_KEYBOARD</td> <td>Installs a keyboard filter. For more information, see the description of the <a href="#">KeyboardProc</a> callback function.</td> </tr> <tr> <td>WH_MOUSE</td> <td>Installs a mouse-message filter. For more information, see the description of the <a href="#">MouseProc</a> callback function.</td> </tr> <tr> <td>WH_MSGFILTER</td> <td>Installs a message filter. For more information, see the description of the <a href="#">MessageProc</a> callback function.</td> </tr> <tr> <td>WH_SHELL</td> <td>Installs a shell-application filter. For more information, see the description of the <a href="#">ShellProc</a> callback function.</td> </tr> <tr> <td>WH_SYSMSGFILTER</td> <td>Installs a system-wide message filter. For more information, see the description of the <a href="#">SysMsgProc</a> callback function.</td> </tr> </tbody> </table> | Value | Meaning | WH_CALLWNDPROC | Installs a window-procedure filter. For more information, see the description of the <a href="#">CallWndProc</a> callback function. | WH_CBT | Installs a computer-based training (CBT) filter. For more information, see the description of the <a href="#">CBTProc</a> callback function. | WH_DEBUG | Installs a debugging filter. For more information, see the description of the <a href="#">DebugProc</a> callback function. | WH_GETMESSAGE | Installs a message filter. For more information, see the description of the <a href="#">GetMsgProc</a> callback function. | WH_HARDWARE | Installs a nonstandard hardware-message filter. For more information, see the description of the <a href="#">HardwareProc</a> callback function. | WH_JOURNALPLAYBACK | Installs a journaling playback filter. For more information, see the description of the <a href="#">JournalPlaybackProc</a> callback function. | WH_JOURNALRECORD | Installs a journaling record filter. For more information, see the description of the <a href="#">JournalRecordProc</a> callback function. | WH_KEYBOARD | Installs a keyboard filter. For more information, see the description of the <a href="#">KeyboardProc</a> callback function. | WH_MOUSE | Installs a mouse-message filter. For more information, see the description of the <a href="#">MouseProc</a> callback function. | WH_MSGFILTER | Installs a message filter. For more information, see the description of the <a href="#">MessageProc</a> callback function. | WH_SHELL | Installs a shell-application filter. For more information, see the description of the <a href="#">ShellProc</a> callback function. | WH_SYSMSGFILTER | Installs a system-wide message filter. For more information, see the description of the <a href="#">SysMsgProc</a> callback function. |
| Value              | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| WH_CALLWNDPROC     | Installs a window-procedure filter. For more information, see the description of the <a href="#">CallWndProc</a> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| WH_CBT             | Installs a computer-based training (CBT) filter. For more information, see the description of the <a href="#">CBTProc</a> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| WH_DEBUG           | Installs a debugging filter. For more information, see the description of the <a href="#">DebugProc</a> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| WH_GETMESSAGE      | Installs a message filter. For more information, see the description of the <a href="#">GetMsgProc</a> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| WH_HARDWARE        | Installs a nonstandard hardware-message filter. For more information, see the description of the <a href="#">HardwareProc</a> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| WH_JOURNALPLAYBACK | Installs a journaling playback filter. For more information, see the description of the <a href="#">JournalPlaybackProc</a> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| WH_JOURNALRECORD   | Installs a journaling record filter. For more information, see the description of the <a href="#">JournalRecordProc</a> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| WH_KEYBOARD        | Installs a keyboard filter. For more information, see the description of the <a href="#">KeyboardProc</a> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| WH_MOUSE           | Installs a mouse-message filter. For more information, see the description of the <a href="#">MouseProc</a> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| WH_MSGFILTER       | Installs a message filter. For more information, see the description of the <a href="#">MessageProc</a> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| WH_SHELL           | Installs a shell-application filter. For more information, see the description of the <a href="#">ShellProc</a> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| WH_SYSMSGFILTER    | Installs a system-wide message filter. For more information, see the description of the <a href="#">SysMsgProc</a> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| hkprc              | Specifies the procedure-instance address of the application-defined hook procedure to be installed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |
| hinst              | Identifies the instance of the module containing the hook function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                |                                                                                                                                     |        |                                                                                                                                              |          |                                                                                                                            |               |                                                                                                                           |             |                                                                                                                                                  |                    |                                                                                                                                                |                  |                                                                                                                                            |             |                                                                                                                              |          |                                                                                                                                |              |                                                                                                                            |          |                                                                                                                                    |                 |                                                                                                                                       |

hTask Identifies the task for which the hook is to be installed. If this parameter is NULL, the installed hook function has system scope and may be called in the context of any process or task in the system.

#### Returns

The return value is a handle of the installed hook, if the function is successful. The application or library must use this handle to identify the hook when it calls the CallNextHookEx and UnhookWindowsHookEx functions. The return value is NULL if an error occurs.

#### Comments

An application or library can use the GetCurrentTask or GetWindowTask function to obtain task handles for use in hooking a particular task.

Hook procedures used with SetWindowsHookEx must be declared as follows:

```
DWORD CALLBACK HookProc(code, wParam, lParam)
int code;
WPARAM wParam;
LPARAM lParam;
{
    if (...)
        return CallNextHookEx(hhook, code, wParam, lParam);
}
```

Chaining to the next hook procedure (that is, calling the CallNextHookProc function) is optional. An application or library can call the next hook procedure either before or after any processing in its own hook procedure.

Before terminating, an application must call the UnhookWindowsHookEx function to free system resources associated with the hook.

Some hooks may be set with system scope only, and others may be set only for a specific task, as shown in the following list:

| Hook               | Scope          |
|--------------------|----------------|
| WH_CALLWNDPROC     | Task or system |
| WH_CBT             | Task or system |
| WH_DEBUG           | Task or system |
| WH_GETMESSAGE      | Task or system |
| WH_HARDWARE        | Task or system |
| WH_JOURNALRECORD   | System only    |
| WH_JOURNALPLAYBACK | System only    |
| WH_KEYBOARD        | Task or system |
| WH_MOUSE           | Task or system |
| WH_MSGFILTER       | Task or system |
| WH_SYSMSGFILTER    | System only    |

For a given hook type, task hooks are called first, then system hooks.

The WH\_CALLWNDPROC hook affects system performance. It is supplied for debugging purposes only.

The system hooks are a shared resource. Installing one affects all applications. All system hook functions must be in libraries. System hooks should be restricted to special-purpose applications or to use as a development aid during debugging of an application. Libraries that no longer need the hook should remove the filter function.

It is a good idea for several reasons to use task hooks rather than system hooks: They do not incur a system-wide overhead in applications that are not affected by the call (or that ignore the call); they do not require packaging the hook-procedure implementation in a separate dynamic-link library; they will continue to work even when future versions of Windows prevent applications from installing system-wide hooks for security reasons.

To install a filter function, the SetWindowsHookEx function must receive a procedure-instance address of the function and the function must be exported in the library's module-definition file. Libraries can pass the procedure address directly. Tasks must use the MakeProcInstance function to get a procedure-instance address. Dynamic-link libraries must use the GetProcAddress function to get a procedure-instance address.

For a given hook type, task hooks are called first, then system hooks.



The WH\_SYSSMSGFILTER hooks are called before the WH\_MSGFILTER hooks. If any of the WH\_SYSSMSGFILTER hook functions return TRUE, the WH\_MSGFILTER hooks are not called.

See Also

CallNextHookEx, GetProcAddress, MakeProcInstance, MessageBox, PeekMessage, PostMessage, SendMessage, UnhookWindowsHookEx

## SetWindowText (2.x)

```
void SetWindowText(hwnd, lpsz)
HWND hwnd;      /* handle of window */
LPCSTR lpsz;    /* address of string */
```

The SetWindowText function sets the given window's title to the specified text.

| Parameter | Description                                                                     |
|-----------|---------------------------------------------------------------------------------|
| hwnd      | Identifies the window or control whose text is to be set.                       |
| lpsz      | Points to a null-terminated string to be used as the new title or control text. |

### Returns

This function does not return a value.

### Comments

This function causes a WM\_SETTEXT message to be sent to the given window or control.

If the window specified by the hwnd parameter is a control, the text within the control is set. If the specified window is a list-box control created with WS\_CAPTION style, however, SetWindowText will set the caption for the control, not for the list-box entries.

### Example

The following example sets a window title:

```
char szBuf[64];
char szFileName[64];

wsprintf((LPSTR) szBuf, "PrntFile - %s", (LPSTR) szFileName);
SetWindowText(hwnd, (LPSTR) szBuf);
```

### See Also

GetWindowText, WM\_SETTEXT

## SetWindowWord (2.x)

WORD SetWindowWord(hwnd, nOffset, nVal)

HWND hwnd; /\* handle of window \*/

int nOffset;

/\* offset of value to set \*/

WORD nVal;

\* new value

/

\*

The SetWindowWord function places a word value at the specified offset into the extra window memory of the given window. Extra window memory is reserved by specifying a nonzero value in the cbWndExtra member of the WNDCLASS structure used with the RegisterClass function.

| Parameter    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |              |                                                                   |       |                                               |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|--------------|-------------------------------------------------------------------|-------|-----------------------------------------------|
| hwnd         | Identifies the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |              |                                                                   |       |                                               |
| nOffset      | Specifies the zero-based byte offset of the value to change. Valid values are in the range zero through the number of bytes of extra window memory, minus two (for example, if 10 or more bytes of extra memory were specified, a value of 8 would be an index to the fifth integer), or one of the following values:<br><table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>GW_HINSTANCE</td><td>Specifies the instance handle of the module that owns the window.</td></tr><tr><td>GW_ID</td><td>Specifies the identifier of the child window.</td></tr></tbody></table> | Value | Meaning | GW_HINSTANCE | Specifies the instance handle of the module that owns the window. | GW_ID | Specifies the identifier of the child window. |
| Value        | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |              |                                                                   |       |                                               |
| GW_HINSTANCE | Specifies the instance handle of the module that owns the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |              |                                                                   |       |                                               |
| GW_ID        | Specifies the identifier of the child window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |              |                                                                   |       |                                               |
| nVal         | Specifies the word value to be placed in the window's reserved memory.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |              |                                                                   |       |                                               |

### Returns

The return value is the previous value of the specified word, if the function is successful. Otherwise, it is zero.

### Comments

To access any extra 2-byte values allocated when the window-class structure was created, use a positive byte offset as the index specified by the nOffset parameter, starting at 0 for the first 2-byte value in the extra space, 2 for the next 2-byte value, and so on.

An application should call the SetParent function, not the SetWindowWord function, to change the parent of a child window.

### See Also

GetWindowLong, GetWindowWord, RegisterClass, SetParent, SetWindowLong

## ShowCaret (2.x)

```
void ShowCaret(hwnd)  
HWND hwnd; /* handle of window with caret */
```

The ShowCaret function shows the caret on the screen at the caret's current position. Once shown, the caret begins flashing automatically.

| Parameter | Description                                                                                                                                            |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window that owns the caret. This parameter can be set to NULL to indirectly specify the window in the current task that owns the caret. |

### Returns

This function does not return a value.

### Comments

The ShowCaret function shows the caret only if it has a current shape and has not been hidden two or more times consecutively. If the given window does not own the caret, the caret is not shown. If the hwnd parameter is NULL, the ShowCaret function shows the caret only if it is owned by a window in the current task.

Hiding the caret is cumulative. If the HideCaret function has been called five times consecutively, ShowCaret must be called five times to show the caret.

The caret is a shared resource. A window should show the caret only when it has the input focus or is active.

### See Also

[CreateCaret](#), [GetActiveWindow](#), [GetFocus](#), [HideCaret](#)

## ShowCursor (2.x)

```
int ShowCursor(fShow)
BOOL fShow; /* cursor visibility flag */
```

The ShowCursor function shows or hides the cursor.

| Parameter | Description                                                                                                                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fShow     | Specifies whether the display count is incremented or decremented (increased or decreased by one). If this parameter is TRUE, the display count is incremented; otherwise, it is decremented. |

### Returns

The return value specifies the new display count, if the function is successful.

### Comments

A cursor show-level count is kept internally. It is incremented by a show operation and decremented by a hide operation. The cursor is visible only if the count is greater than or equal to zero. If a mouse exists, the initial setting of the cursor show level is zero; otherwise, it is -1.

The cursor is a shared resource. A window that hides the cursor should show it before the cursor leaves its client area or before the window relinquishes control to another window.

### See Also

[SetCursor](#)

## ShowOwnedPopups (2.x)

```
void ShowOwnedPopups(HWND hwnd, BOOL fShow)
/* handle of window */
/* window visibility flag */
```

The ShowOwnedPopups function shows or hides all pop-up windows owned by the given window.

| Parameter | Description                                                                                                                                                                                    |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window that owns the pop-up windows to be shown or hidden.                                                                                                                      |
| fShow     | Specifies whether pop-up windows are to be shown or hidden. If this parameter is TRUE, all hidden pop-up windows are shown. If this parameter is FALSE, all visible pop-up windows are hidden. |

### Returns

This function does not return a value.

### See Also

[IsWindowVisible](#), [ShowWindow](#)

## ShowScrollBar (2.x)

```
void ShowScrollBar(hwnd, fnBar, fShow)
HWND hwnd;      /* handle of window with scroll bar */
int fnBar;      /* scroll-bar flag */
BOOL fShow;
/* scroll-bar visibility flag
/
```

The ShowScrollBar function shows or hides a scroll bar.

| Parameter            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                      |                                                             |                     |                                                                    |                      |                                               |                      |                                             |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|----------------------|-------------------------------------------------------------|---------------------|--------------------------------------------------------------------|----------------------|-----------------------------------------------|----------------------|---------------------------------------------|
| hwnd                 | Identifies a scroll bar or a window that contains a scroll bar in its nonclient area, depending on the value of the fnBar parameter. If fnBar is <code>SB_CTL</code> , hwnd identifies a scroll bar. If fnBar is <code>SB_HORZ</code> , <code>SB_VERT</code> , or <code>SB_BOTH</code> , hwnd identifies a window that has a scroll bar in its nonclient area.                                                                                                                                                                                                                                                                                                                                                                                                                               |       |         |                      |                                                             |                     |                                                                    |                      |                                               |                      |                                             |
| fnBar                | Specifies whether the scroll bar is a control or part of a window's nonclient area. If the scroll bar is part of the nonclient area, fnBar also indicates whether the scroll bar is positioned horizontally, vertically, or both. This parameter can be one of the following values:<br><table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td><code>SB_BOTH</code></td><td>Specifies the window's horizontal and vertical scroll bars.</td></tr><tr><td><code>SB_CTL</code></td><td>Specifies that the hwnd parameter identifies a scroll bar control.</td></tr><tr><td><code>SB_HORZ</code></td><td>Specifies the window's horizontal scroll bar.</td></tr><tr><td><code>SB_VERT</code></td><td>Specifies the window's vertical scroll bar.</td></tr></tbody></table> | Value | Meaning | <code>SB_BOTH</code> | Specifies the window's horizontal and vertical scroll bars. | <code>SB_CTL</code> | Specifies that the hwnd parameter identifies a scroll bar control. | <code>SB_HORZ</code> | Specifies the window's horizontal scroll bar. | <code>SB_VERT</code> | Specifies the window's vertical scroll bar. |
| Value                | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                      |                                                             |                     |                                                                    |                      |                                               |                      |                                             |
| <code>SB_BOTH</code> | Specifies the window's horizontal and vertical scroll bars.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                      |                                                             |                     |                                                                    |                      |                                               |                      |                                             |
| <code>SB_CTL</code>  | Specifies that the hwnd parameter identifies a scroll bar control.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                      |                                                             |                     |                                                                    |                      |                                               |                      |                                             |
| <code>SB_HORZ</code> | Specifies the window's horizontal scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                      |                                                             |                     |                                                                    |                      |                                               |                      |                                             |
| <code>SB_VERT</code> | Specifies the window's vertical scroll bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                      |                                                             |                     |                                                                    |                      |                                               |                      |                                             |
| fShow                | Specifies whether the scroll bar is shown or hidden. If this parameter is <code>TRUE</code> , the scroll bar is shown; otherwise, it is hidden.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                      |                                                             |                     |                                                                    |                      |                                               |                      |                                             |

### Returns

This function does not return a value.

### Comments

An application should not call this function to hide a scroll bar while processing a scroll-bar notification message.

### See Also

[GetScrollPos](#), [GetScrollRange](#), [ScrollWindow](#), [SetScrollPos](#), [SetScrollRange](#)

## ShowWindow (2.x)

```
BOOL ShowWindow(HWND hwnd, int nCmdShow)
/* handle of window */
/* window visibility flag */
```

The ShowWindow function sets the given window's visibility state.

| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|---------|-----------------------------------------------------------|-------------|-----------------------------------------------------------------------------------------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|---------|----------------------------------------------------------------------|------------------|-----------------------------------------------------------|------------------|------------------------------------------------|--------------------|-----------------------------------------------------------------------------------|-----------|---------------------------------------------------------------------------------------------|-------------------|-------------------------------------------------------------------------------------------------------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd               | Identifies the window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |
| nCmdShow           | Specifies how the window is to be shown. This parameter can be one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |
|                    | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>SW_HIDE</td><td>Hides the window and passes activation to another window.</td></tr><tr><td>SW_MINIMIZE</td><td>Minimizes the specified window and activates the top-level window in the system's list.</td></tr><tr><td>SW_RESTORE</td><td>Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_SHOWNORMAL).</td></tr><tr><td>SW_SHOW</td><td>Activates a window and displays it in its current size and position.</td></tr><tr><td>SW_SHOWMAXIMIZED</td><td>Activates a window and displays it as a maximized window.</td></tr><tr><td>SW_SHOWMINIMIZED</td><td>Activates a window and displays it as an icon.</td></tr><tr><td>SW_SHOWMINNOACTIVE</td><td>Displays a window as an icon. The window that is currently active remains active.</td></tr><tr><td>SW_SHOWNA</td><td>Displays a window in its current state. The window that is currently active remains active.</td></tr><tr><td>SW_SHOWNOACTIVATE</td><td>Displays a window in its most recent size and position. The window that is currently active remains active.</td></tr><tr><td>SW_SHOWNORMAL</td><td>Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_RESTORE).</td></tr></tbody></table> | Value | Meaning | SW_HIDE | Hides the window and passes activation to another window. | SW_MINIMIZE | Minimizes the specified window and activates the top-level window in the system's list. | SW_RESTORE | Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_SHOWNORMAL). | SW_SHOW | Activates a window and displays it in its current size and position. | SW_SHOWMAXIMIZED | Activates a window and displays it as a maximized window. | SW_SHOWMINIMIZED | Activates a window and displays it as an icon. | SW_SHOWMINNOACTIVE | Displays a window as an icon. The window that is currently active remains active. | SW_SHOWNA | Displays a window in its current state. The window that is currently active remains active. | SW_SHOWNOACTIVATE | Displays a window in its most recent size and position. The window that is currently active remains active. | SW_SHOWNORMAL | Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_RESTORE). |
| Value              | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |
| SW_HIDE            | Hides the window and passes activation to another window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |
| SW_MINIMIZE        | Minimizes the specified window and activates the top-level window in the system's list.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |
| SW_RESTORE         | Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_SHOWNORMAL).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |
| SW_SHOW            | Activates a window and displays it in its current size and position.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |
| SW_SHOWMAXIMIZED   | Activates a window and displays it as a maximized window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |
| SW_SHOWMINIMIZED   | Activates a window and displays it as an icon.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |
| SW_SHOWMINNOACTIVE | Displays a window as an icon. The window that is currently active remains active.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |
| SW_SHOWNA          | Displays a window in its current state. The window that is currently active remains active.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |
| SW_SHOWNOACTIVATE  | Displays a window in its most recent size and position. The window that is currently active remains active.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |
| SW_SHOWNORMAL      | Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_RESTORE).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |         |                                                           |             |                                                                                         |            |                                                                                                                                                          |         |                                                                      |                  |                                                           |                  |                                                |                    |                                                                                   |           |                                                                                             |                   |                                                                                                             |               |                                                                                                                                                       |

### Returns

The return value is nonzero if the window was previously visible. It is zero if the window was previously hidden.

### Comments

The ShowWindow function must be called only once per application using the nCmdShow parameter from the WinMain function. Subsequent calls to ShowWindow must use one of the values listed in the preceding list, instead of the one specified by the nCmdShow parameter from WinMain.

### See Also

IsWindowVisible, ShowOwnedPopups, WM\_SHOWWINDOW



### SubtractRect (3.1)

```
BOOL SubtractRect(lprcDest, lprcSource1, lprcSource2)
RECT FAR* lprcDest;          /* pointer to destination rectangle */
const RECT FAR* lprcSource1; /* pointer to rect. to subtract from */
/
const RECT FAR* lprcSource2; /* pointer to rect. to subtract */
/
```

The SubtractRect function retrieves the coordinates of a rectangle by subtracting one rectangle from another.

| Parameter   | Description                                                                                                              |
|-------------|--------------------------------------------------------------------------------------------------------------------------|
| lprcDest    | Points to the <u>RECT</u> structure to receive the dimensions of the new rectangle.                                      |
| lprcSource1 | Points to the <u>RECT</u> structure from which a rectangle is to be subtracted.                                          |
| lprcSource2 | Points to the <u>RECT</u> structure that is to be subtracted from the rectangle pointed to by the lprcSource1 parameter. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The rectangle specified by the lprcSource2 parameter is subtracted from the rectangle specified by lprcSource1 only when the rectangles intersect completely in either the x- or y-direction. For example, if lprcSource1 were (10,10, 100,100) and lprcSource2 were (50,50, 150,150), the rectangle pointed to by lprcDest would contain the same coordinates as lprcSource1 when the function returned. If lprcSource1 were (10,10, 100,100) and lprcSource2 were (50,10, 150,150), however, the rectangle pointed to by lprcDest would contain the coordinates (10,10, 50,100) when the function returned.

#### See Also

IntersectRect, UnionRect, RECT

## SwapMouseButton (2.x)

```
BOOL SwapMouseButton(fSwap)
BOOL fSwap; /* reverse or restore buttons */
```

The SwapMouseButton function reverses the meaning of left and right mouse buttons.

| Parameter | Description                                                                                                                                                                                                                                                                            |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fSwap     | Specifies whether the button meanings are reversed or restored. If this parameter is TRUE, the left button generates right-button mouse messages and the right button generates left-button messages. If this parameter is FALSE, the buttons are restored to their original meanings. |

### Returns

The return value specifies the meaning of the mouse buttons immediately before the function is called. It is nonzero if the meaning was reversed. Otherwise, it is zero.

### Comments

Button swapping is provided as a convenience to people who use the mouse with their left hands. The SwapMouseButton function is usually called by Control Panel only. Although an application is free to call the function, the mouse is a shared resource and reversing the meaning of the mouse button affects all applications.

### Example

The following example swaps the mouse buttons, depending on the check state of a check box:

```
BOOL fSwap;

fSwap = (BOOL) SendMessage(hdlg, IDD_SWAP,
BM_GETCHECK, 0, 0L);
SwapMouseButton(fSwap);
```

### SystemParametersInfo (3.1)

```

BOOL SystemParametersInfo(uAction, uParam, lpvParam, fuWinIni)
UINT uAction;          /* system parameter to query or set */
UINT uParam;          /* depends on
system parameter
/
void FAR* lpvParam;
* depends on system parameter
/
UINT fuWinIni;
* WIN.INI update flag
/

```

The SystemParametersInfo function queries or sets system-wide parameters. This function can also update the WIN.INI file while setting a parameter.

| Parameter | Description                                                                                             | Meaning                                                                                                        |
|-----------|---------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| uAction   | Specifies the system-wide parameter to query or set. This parameter can be one of the following values: |                                                                                                                |
|           | Value                                                                                                   |                                                                                                                |
|           | <u>SPI_GETBEEP</u>                                                                                      | Retrieves a <b>BOOL</b> value that indicates whether the warning beep is on or off.                            |
|           | <u>SPI_GETBORDER</u>                                                                                    | Retrieves the border multiplying factor that determines the width of a window's sizing border.                 |
|           | <u>SPI_GETFASTTASKSWITCH</u>                                                                            | Determines whether fast task switching is on or off.                                                           |
|           | <u>SPI_GETGRIDGRANULARITY</u>                                                                           | Retrieves the current granularity value of the desktop sizing grid.                                            |
|           | <u>SPI_GETICONTITLELOGFONT</u>                                                                          | Retrieves the logical-font information for the current icon-title font.                                        |
|           | <u>SPI_GETICONTITLEWRAP</u>                                                                             | Determines whether icon-title wrapping is on or off.                                                           |
|           | <u>SPI_GETKEYBOARDDELAY</u>                                                                             | Retrieves the keyboard repeat-delay setting.                                                                   |
|           | <u>SPI_GETKEYBOARDSPEED</u>                                                                             | Retrieves the keyboard repeat-speed setting.                                                                   |
|           | <u>SPI_GETMENUDROPALIGNMENT</u>                                                                         | Determines whether pop-up menus are left-aligned or right-aligned relative to the corresponding menu-bar item. |
|           | <u>SPI_GETMOUSE</u>                                                                                     | Retrieves the mouse speed and the mouse threshold values, which Windows uses to calculate mouse acceleration.  |
|           | <u>SPI_GETSCREENSAVEACTIVE</u>                                                                          | Retrieves a <b>BOOL</b> value that indicates whether screen saving is on or off.                               |
|           | <u>SPI_GETSCREENSAVETIMEOUT</u>                                                                         | Retrieves the screen-saver time-out value.                                                                     |
|           | <u>SPI_ICONHORIZONTALSPACING</u>                                                                        | Sets the width, in pixels, of an icon cell.                                                                    |
|           | <u>SPI_ICONVERTICALSPACING</u>                                                                          | Sets the height, in pixels, of an icon cell.                                                                   |
|           | <u>SPI_LANGDRIVER</u>                                                                                   | Forces the user to load a new language driver.                                                                 |

|                                 |                                                                                                                                                                             |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>SPI_SETBEEP</u>              | Turns the warning beep on or off.                                                                                                                                           |
| <u>SPI_SETBORDER</u>            | Sets the border multiplying factor that determines the width of a window's sizing border.                                                                                   |
| <u>SPI_SETDESKPATTERN</u>       | Sets the current desktop pattern to the value specified in the Pattern entry in the WIN.INI file or to the pattern specified by the lpvParam parameter.                     |
| <u>SPI_SETDESKWALLPAPER</u>     | Specifies the filename that contains the bitmap to be used as the desktop wallpaper.                                                                                        |
| <u>SPI_SETDOUBLECLKHEIGHT</u>   | Sets the height of the rectangle within which the second click of a double-click must fall for it to be registered as a double-click.                                       |
| <u>SPI_SETDOUBLECLICKTIME</u>   | Sets the double-click time for the mouse. The double-click time is the maximum number of milliseconds that may occur between the first and second clicks of a double-click. |
| <u>SPI_SETDOUBLECLKWIDTH</u>    | Sets the width of the rectangle within which the second click of a double-click must fall for it to be registered as a double-click.                                        |
| <u>SPI_SETFASTTASKSWITCH</u>    | Turns fast task switching on or off.                                                                                                                                        |
| <u>SPI_SETGRIDGRANULARITY</u>   | Sets the granularity of the desktop sizing grid.                                                                                                                            |
| <u>SPI_SETICONTITLELOGFONT</u>  | Sets the font that is used for icon titles.                                                                                                                                 |
| <u>SPI_SETICONTITLEWRAP</u>     | Turns icon-title wrapping on or off.                                                                                                                                        |
| <u>SPI_SETKEYBOARDDELAY</u>     | Sets the keyboard repeat-delay setting.                                                                                                                                     |
| <u>SPI_SETKEYBOARDSPEED</u>     | Sets the keyboard repeat-speed setting.                                                                                                                                     |
| <u>SPI_SETMENUDROPALIGNMENT</u> | Sets the alignment value of pop-up menus.                                                                                                                                   |
| <u>SPI_SETMOUSE</u>             | Sets the mouse speed and the x and y mouse-threshold values.                                                                                                                |
| <u>SPI_SETMOUSEBUTTONSWAP</u>   | Swaps or restores the meaning of the left and right mouse buttons.                                                                                                          |
| <u>SPI_SETSCREENSAVEACTIVE</u>  | Sets the state of the screen saver.                                                                                                                                         |
| <u>SPI_SETSCREENSAVETIMEOUT</u> | Sets the screen-saver time-out value.                                                                                                                                       |

uParam Depends on the uAction parameter. For more information, see the following Comments section.

lpvParam Depends on the uAction parameter. For more information, see the following Comments section.

fuWinIni

If a system parameter is being set, specifies whether the WIN.INI file is updated, and if so, whether the WM\_WININICHANGE message is broadcast to all top-level windows to notify them of the change. This parameter can be a combination of the following values:

| Value                        | Meaning                                                                                                                                              |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>SPIF_UPDATEINIFILE</u>    | Writes the new system-wide parameter setting to the WIN.INI file.                                                                                    |
| <u>SPIF_SENDWININICHANGE</u> | Broadcasts the <u>WM_WININICHANGE</u> message if the WIN.INI file is updated. This flag has no effect if <u>SPIF_UPDATEINIFILE</u> is not specified. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The SystemParameterInfo function is intended for applications, such as Control Panel, that allow the user to customize the Windows environment.

The following table describes the uParam and lpvParam parameters for each SPI\_ constant:

| Constant                 | uParam                           | lpvParam                                                                                                                                                                                               |
|--------------------------|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SPI_GETBEEP              | 0                                | Points to a BOOL variable that receives TRUE if the beep is on, FALSE if it is off.                                                                                                                    |
| SPI_GETBORDER            | 0                                | Points to an integer variable that receives the border multiplying factor.                                                                                                                             |
| SPI_GETFASTTASKSWITCH    | 0                                | Points to a BOOL variable that receives TRUE if fast task switching is on, FALSE if it is off.                                                                                                         |
| SPI_GETGRIDGRANULARITY   | 0                                | Points to an integer variable that receives the grid-granularity value.                                                                                                                                |
| SPI_GETICONTITLELOGFONT  | Size of <u>LOGFONT</u> structure | Points to a <u>LOGFONT</u> structure that receives the logical-font information.                                                                                                                       |
| SPI_GETICONTITLEWRAP     | 0                                | Points to a BOOL variable that receives TRUE if wrapping is on, FALSE if wrapping is off.                                                                                                              |
| SPI_GETKEYBOARDDELAY     | 0                                | Points to an integer variable that receives the keyboard repeat-delay setting.                                                                                                                         |
| SPI_GETKEYBOARDSPEED     | 0                                | Points to a <u>WORD</u> variable that receives the current keyboard repeat-speed setting.                                                                                                              |
| SPI_GETMENUDROPALIGNMENT | 0                                | Points to a BOOL variable that receives TRUE if pop-up menus are right-aligned, FALSE if they are left-aligned.                                                                                        |
| SPI_GETMOUSE             | 0                                | Points to an integer array name lpiMouse, where lpiMouse[0] receives the WIN.INI entry MouseThreshold1, lpiMouse[1] receives the entry MouseThreshold2, and lpiMouse[2] receives the entry MouseSpeed. |
| SPI_GETSCREENSAVEACTIVE  | 0                                | Points to a BOOL variable that receives TRUE if the screen saver is active, FALSE if it is                                                                                                             |

|                           |                                                         |                                                                                                                                                                                                                                                                                                                                              |
|---------------------------|---------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SPI_GETSCREENSAVETIMEOUT  | 0                                                       | not.<br>Points to an integer variable that receives the screen-saver time-out value, in milliseconds.                                                                                                                                                                                                                                        |
| SPI_ICONHORIZONTALSPACING | New width, in pixels, for horizontal spacing of icons   | Is NULL if the icon cell width, in pixels, is returned in uParam. If this value is a pointer to an integer, the current horizontal spacing is returned in that variable and uParam is ignored.                                                                                                                                               |
| SPI_ICONVERTICALSPACING   | New height, in pixels, for vertical spacing of icons    | Is NULL if the icon cell height, in pixels, is returned in uParam. If this value is a pointer to an integer, the current vertical spacing is returned in that variable and uParam is ignored.                                                                                                                                                |
| SPI_LANGDRIVER            | 0                                                       | Points to a string containing the new language driver filename. The application should make sure that all other international settings remain consistent when changing the language driver.                                                                                                                                                  |
| SPI_SETBEEP               | TRUE = turn the beep on; FALSE = turn the beep off      | Is NULL.                                                                                                                                                                                                                                                                                                                                     |
| SPI_SETBORDER             | Border multiplying factor                               | Is NULL.                                                                                                                                                                                                                                                                                                                                     |
| SPI_SETDESKPATTERN        | 0 or -1                                                 | Specifies the desktop pattern. If this value is NULL and the uParam parameter is -1, the value is reread from the WIN.INI file. This value can also be a null-terminated string (LPSTR) containing a sequence of 8 numbers that represent the new desktop pattern; for example, "170 85 170 85 170 85 170 85" represents a 50% gray pattern. |
| SPI_SETDESKWALLPAPER      | 0                                                       | Points to a string that specifies the name of the bitmap file.                                                                                                                                                                                                                                                                               |
| SPI_SETDOUBLECLKHEIGHT    | Double-click height, in pixels                          | Is NULL.                                                                                                                                                                                                                                                                                                                                     |
| SPI_SETDOUBLECLICKTIME    | Double-click time, in milliseconds                      | Is NULL.                                                                                                                                                                                                                                                                                                                                     |
| SPI_SETDOUBLECLKWIDTH     | Double-click width, in pixels                           | Is NULL.                                                                                                                                                                                                                                                                                                                                     |
| SPI_SETFASTTASKSWITCH     | TRUE = turn on fast task switching; FALSE = turn it off | Is NULL.                                                                                                                                                                                                                                                                                                                                     |
| SPI_SETGRIDGRANULARITY    | Grid granularity                                        |                                                                                                                                                                                                                                                                                                                                              |
| SPI_SETICONTITLELOGFONT   | Size of the <u>LOGFONT</u> structure                    | Points to a <u>LOGFONT</u> structure that defines the                                                                                                                                                                                                                                                                                        |

|                          |                                                                                                                        |                                                                                                                                                                                                         |
|--------------------------|------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |                                                                                                                        | font to use for icon titles. If uParam is set to zero and lParam is set to NULL, Windows uses the icon-title font and spacings that were in effect when Windows was started.                            |
| SPI_SETICONTITLEWRAP     | TRUE = turn wrapping on; FALSE = turn wrapping off                                                                     | Is NULL.                                                                                                                                                                                                |
| SPI_SETKEYBOARDDELAY     | Keyboard-delay setting                                                                                                 | Is NULL.                                                                                                                                                                                                |
| SPI_SETKEYBOARDSPEED     | Repeat-speed setting                                                                                                   | Is NULL.                                                                                                                                                                                                |
| SPI_SETMENUDROPALIGNMENT | TRUE = right-alignment; FALSE = left-alignment                                                                         | Is NULL.                                                                                                                                                                                                |
| SPI_SETMOUSE             | 0                                                                                                                      | Points to an integer array named lpiMouse, where lpiMouse[0] receives the WIN.INI entry xMouseThreshold, lpiMouse[1] receives the entry yMouseThreshold, and lpiMouse[2] receives the entry MouseSpeed. |
| SPI_SETMOUSEBUTTONSWAP   | TRUE = reverse the meaning of the left and right mouse buttons; FALSE = restore the buttons to their original meanings | Is NULL.                                                                                                                                                                                                |
| SPI_SETSCREENSAVEACTIVE  | TRUE = activate screen saving; FALSE = deactivate screen saving                                                        | Is NULL.                                                                                                                                                                                                |
| SPI_SETSCREENSAVETIMEOUT | Idle time-out duration, in seconds, before screen is saved                                                             | Is NULL.                                                                                                                                                                                                |

#### Example

The following example retrieves the value for the DoubleClickSpeed entry from the WIN.INI file and uses the value to initialize an edit control. In this example, while the WM\_COMMAND message is being processed, the user-specified value is retrieved from the edit control and used to set the double-click time.

```
char szBuf[32];
int iResult;

case WM_INITDIALOG:
    /* Initialize edit control to the current double-click time. */

    iResult = GetProfileInt("windows",
    "DoubleClickSpeed", 550);
    itoa(iResult, szBuf, 10);
    SendDlgItemMessage(hdlg, IDD_DCLKTIME, WM_SETTEXT, 0,
    (DWORD) (LPSTR) szBuf);

    .
    . /* Initialize any other controls. */
    .

    return 0;

case WM_COMMAND:
    switch(wParam) {

        case IDOK:

            /* Set double-click time to a user-specified value. */
```

```
SendDlgItemMessage(hdlg, IDD_DCLKTIME, WM_GETTEXT,  
sizeof(szBuf), (DWORD) (LPSTR) szBuf);  
SystemParametersInfo(SPI_SETDOUBLECLICKTIME, atoi(szBuf),  
(LPVOID) NULL, SPIF_UPDATEINIFILE |  
SPIF_SENDWININICHANGE);  
  
. /* Set any other system-wide parameters. */  
.  
  
EndDialog(hdlg, TRUE);  
return TRUE;  
}  
return 0;
```

See Also

WM\_WININICHANGE



SPI\_GETBEEP 1

Retrieves a BOOL value that indicates whether the warning beep is on or off.

SPI\_GETBEEP 1

SPI\_GETBORDER 5

Retrieves the border multiplying factor that determines the width of a window's sizing border.

SPI\_GETBORDER 5

SPI\_GETFASTTASKSWITCH 35

Determines whether fast task switching is on or off.



SPI\_GETGRIDGRANULARITY 18

Retrieves the current granularity value of the desktop sizing grid.

SPI\_GETGRIDGRANULARITY 18



SPI\_GETICONTITLELOGFONT 31

Retrieves the logical-font information for the current icon-title font.

SPI\_GETICONTITLELOGFONT 31

SPI\_GETICONTITLEWRAP 25

Determines whether icon-title wrapping is on or off.



SPI\_GETKEYBOARDDELAY 22

Retrieves the keyboard repeat-delay setting.

SPI\_GETKEYBOARDDELAY 22

SPI\_GETKEYBOARDSPEED 10

Retrieves the keyboard repeat-speed setting.

SPI\_GETKEYBOARDSPEED 10



`SPI_GETMENUDROPALIGNMENT` 27

Determines whether pop-up menus are left-aligned or right-aligned relative to the corresponding menu-bar item.

SPI\_GETMENUDROPALIGNMENT 27

### SPI\_GETMOUSE 3

Retrieves the mouse speed and the mouse threshold values, which Windows uses to calculate mouse acceleration.

SPI\_GETMOUSE 3

SPI\_GETSCREENSAVEACTIVE 16

Retrieves a BOOL value that indicates whether screen saving is on or off.

SPI\_GETSCREENSAVEACTIVE 16

SPI\_GETSCREENSAVETIMEOUT 14  
Retrieves the screen-saver time-out value.

SPI\_GETSCREENSAVETIMEOUT 14



SPI\_ICONHORIZONTALSPACING 13

Sets the width, in pixels, of an icon cell.

SPI\_ICONHORIZONTALSPACING 13

SPI\_ICONVERTICALSPACING 24  
Sets the height, in pixels, of an icon cell.

SPI\_CONVERTICALSPACING 24

SPI\_LANGDRIVER 12

Forces the user to load a new language driver.

SPI\_LANGDRIVER 12

SPI\_SETBEEP 2

Turns the warning beep on or off.

SPI\_SETBEEP 2



`SPI_SETBORDER` 6

Sets the border multiplying factor that determines the width of a window's sizing border.

SPI\_SETBORDER 6

**SPI\_SETDESKPATTERN 21**

Sets the current desktop pattern to the value specified in the Pattern entry in the WIN.INI file or to the pattern specified by the lpvParam parameter.

SPI\_SETDESKPATTERN 21

SPI\_SETDESKWALLPAPER 20

Specifies the filename that contains the bitmap to be used as the desktop wallpaper.

SPI\_SETDESKWALLPAPER 20

`SPI_SETDOUBLECLKHEIGHT 30`

Sets the height of the rectangle within which the second click of a double-click must fall for it to be registered as a double-click.

SPI\_SETDOUBLECLKHEIGHT 30



`SPI_SETDOUBLECLICKTIME` 32

Sets the double-click time for the mouse. The double-click time is the maximum number of milliseconds that may occur between the first and second clicks of a double-click.

SPI\_SETDOUBLECLICKTIME 32

`SPI_SETDOUBLECLKWIDTH` 29

Sets the width of the rectangle within which the second click of a double-click must fall for it to be registered as a double-click.

SPI\_SETDOUBLECLKWIDTH 29

SPI\_SETFASTTASKSWITCH 36

Turns fast task switching on or off.



SPI\_SETGRIDGRANULARITY 19

Sets the granularity of the desktop sizing grid.

SPI\_SETGRIDGRANULARITY 19



SPI\_SETICONTITLELOGFONT 34  
Sets the font that is used for icon titles.

SPI\_SETCONTITLELOGFONT 34

SPI\_SETICONTITLEWRAP 26  
Turns icon-title wrapping on or off.



SPI\_SETKEYBOARDDELAY 23

Sets the keyboard repeat-delay setting.

SPI\_SETKEYBOARDDELAY 23

SPI\_SETKEYBOARDSPEED 11

Sets the keyboard repeat-speed setting.

SPI\_SETKEYBOARDSPEED 11



SPI\_SETMENUDROPALIGNMENT 28  
Sets the alignment value of pop-up menus.

SPI\_SETMENUMDROALIGNMENT 28

SPI\_SETMOUSE 4

Sets the mouse speed and the x and y mouse-threshold values.

SPI\_SETMOUSE 4

`SPI_SETMOUSEBUTTONSWAP` 33

Swaps or restores the meaning of the left and right mouse buttons.

SPI\_SETMOUSEBUTTONSWAP 33

SPI\_SETSCREENSAVEACTIVE 17

Sets the state of the screen saver.

SPI\_SETSCREENSAVEACTIVE 17



SPI\_SETSCREENSAVETIMEOUT 15

Sets the screen-saver time-out value.

SPI\_SETSCREENSAVETIMEOUT 15

SPIF\_UPDATEINIFILE 0x0001

Writes the new system-wide parameter setting to the WIN.INI file.

SPIF\_UPDATEINIFILE 0x0001

SPIF\_SENDWININICHANGE 0x0002

Broadcasts the WM\_WININICHANGE message if the WIN.INI file is updated. This flag has no effect if SPIF\_UPDATEINIFILE is not specified.

SPIF\_SENDWININICHANGE 0x0002

### TabbedTextOut (3.0)

```
LONG TabbedTextOut(hdc, xPosStart, yPosStart, lpszString, cbString, cTabStops, lpnTabPositions,
                  nTabOrigin)
HDC hdc; /* handle of device context */
int xPosStart; /* x-
coordinate of starting position
/
int yPosStart;
* y-coordinate of starting position
/
LPCSTR lpszString;
* address of string
/
int cbString;
* number of characters in string
/
int cTabStops;
* number of tabs in array
/
int FAR* lpnTabPositions;
* address of array with tab positions
/
int nTabOrigin;
* x-coordinate for tab expansion
/
```

The TabbedTextOut function writes a character string at the specified location, expanding tabs to the values specified in the array of tab-stop positions. The function writes text in the currently selected font.

| Parameter       | Description                                                                                                                                                                          |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hdc             | Identifies the device context.                                                                                                                                                       |
| xPosStart       | Specifies the logical x-coordinate of the starting point of the string.                                                                                                              |
| yPosStart       | Specifies the logical y-coordinate of the starting point of the string.                                                                                                              |
| lpszString      | Points to the character string to be drawn.                                                                                                                                          |
| cbString        | Specifies the number of characters in the string.                                                                                                                                    |
| cTabStops       | Specifies the number of values in the array of tab-stop positions.                                                                                                                   |
| lpnTabPositions | Points to an array containing the tab-stop positions, in device units. The tab stops must be sorted in increasing order; the smallest x-value should be the first item in the array. |
| nTabOrigin      | Specifies the logical x-coordinate of the starting position from which tabs are expanded.                                                                                            |

#### Returns

The return value is the dimensions of the string, in logical units, if the function is successful. The low-order word contains the string width; the high-order word contains the string height. Otherwise, the return value is zero.

#### Comments

If the cTabStops parameter is zero and the lpnTabPositions parameter is NULL, tabs are expanded to eight times the average character width.

If cTabStops is 1, the tab stops are separated by the distance specified by the first value in the lpnTabPositions array.

If the lpnTabPositions array contains more than one value, a tab stop is set for each value in the array, up to the number specified by cTabStops.

The nTabOrigin parameter allows an application to call the TabbedTextOut function several times for a single line. If the application calls TabbedTextOut more than once with the nTabOrigin set to the same value each time, the function expands all tabs relative to the position specified by nTabOrigin.

By default, the current position is not used or updated by the TabbedTextOut function. If an application must update the current position when calling TabbedTextOut, it can call the SetTextAlign function with the wFlags parameter set to TA\_UPDATECP. When this flag is set, Windows ignores the xPosStart and

yPosStart parameters on subsequent calls to the TabbedTextOut function, using the current position instead.

#### Example

The following example expands tabs from the same x-coordinate as the string's starting point:

```
LPSTR lpszTabbedText = "Column 1\tColumn 2\tTest of TabbedTextOut";  
int aTabs[2] = { 150, 300 };  
int iStartXPos = 100;  
int iStartYPos = 100;
```

```
TabbedTextOut(hdc, /* handle of device context */  
              iStartXPos, iStartYPos, /* starting coordinates */  
              lpszTabbedText, /* address of text */  
              strlen(lpszTabbedText), /* number of characters */  
              sizeof(aTabs) / sizeof(int), /* number of tabs in array */  
              aTabs, /* array for tab positions */  
              iStartXPos); /* x-coord. for tab expanding */
```

#### See Also

[GetTabbedTextExtent](#), [SetTextAlign](#), [SetTextColor](#), [TextOut](#)



### TrackPopupMenu (3.0)

```

BOOL TrackPopupMenu(hmenu, fuFlags, x, y, nReserved, hwnd, lprc)
HMENU hmenu;           /* handle of menu */
UINT fuFlags;          /* screen-position and mouse-button flags */
/
int x;
/* horizontal screen position */
/
int y;
/* vertical screen position */
/
int nReserved;
/* reserved */
/
HWND hwnd;
/* handle of owner window */
/
const RECT FAR* lprc;
/* address of structure with rectangle */
/

```

The TrackPopupMenu function displays the given floating pop-up menu at the specified location and tracks the selection of items on the pop-up menu. A floating pop-up menu can appear anywhere on the screen.

| Parameter              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|------------------------|-----------------------------------------------------------------------------------------------|----------------------|--------------------------------------------------------------------------------------------------------------|-----------------------|---------------------------------------------------------------------------------------------------------------|-------|---------|-----------------------|--------------------------------------------------------|------------------------|-----------------------------------------------------------------------------|
| hmenu                  | Identifies the pop-up menu to be displayed. The application retrieves this handle by calling the <u>CreatePopupMenu</u> function to create a new pop-up menu or by calling the <u>GetSubMenu</u> function to retrieve the handle of a pop-up menu associated with an existing menu item.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
| fuFlags                | Specifies the screen-position and mouse-button flags. The screen-position flag can be one of the following: <table border="0" style="margin-left: 20px;"> <thead> <tr> <th style="text-align: left;">Value</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td><u>TPM_CENTERALIGN</u></td> <td>Centers the pop-up menu horizontally relative to the coordinate specified by the x parameter.</td> </tr> <tr> <td><u>TPM_LEFTALIGN</u></td> <td>Positions the pop-up menu so that its left side is aligned with the coordinate specified by the x parameter.</td> </tr> <tr> <td><u>TPM_RIGHTALIGN</u></td> <td>Positions the pop-up menu so that its right side is aligned with the coordinate specified by the x parameter.</td> </tr> </tbody> </table> <p>The mouse-button flag can be one of the following:</p> <table border="0" style="margin-left: 20px;"> <thead> <tr> <th style="text-align: left;">Value</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td><u>TPM_LEFTBUTTON</u></td> <td>Causes the pop-up menu to track the left mouse button.</td> </tr> <tr> <td><u>TPM_RIGHTBUTTON</u></td> <td>Causes the pop-up menu to track the right mouse button instead of the left.</td> </tr> </tbody> </table> | Value | Meaning | <u>TPM_CENTERALIGN</u> | Centers the pop-up menu horizontally relative to the coordinate specified by the x parameter. | <u>TPM_LEFTALIGN</u> | Positions the pop-up menu so that its left side is aligned with the coordinate specified by the x parameter. | <u>TPM_RIGHTALIGN</u> | Positions the pop-up menu so that its right side is aligned with the coordinate specified by the x parameter. | Value | Meaning | <u>TPM_LEFTBUTTON</u> | Causes the pop-up menu to track the left mouse button. | <u>TPM_RIGHTBUTTON</u> | Causes the pop-up menu to track the right mouse button instead of the left. |
| Value                  | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
| <u>TPM_CENTERALIGN</u> | Centers the pop-up menu horizontally relative to the coordinate specified by the x parameter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
| <u>TPM_LEFTALIGN</u>   | Positions the pop-up menu so that its left side is aligned with the coordinate specified by the x parameter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
| <u>TPM_RIGHTALIGN</u>  | Positions the pop-up menu so that its right side is aligned with the coordinate specified by the x parameter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
| Value                  | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
| <u>TPM_LEFTBUTTON</u>  | Causes the pop-up menu to track the left mouse button.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
| <u>TPM_RIGHTBUTTON</u> | Causes the pop-up menu to track the right mouse button instead of the left.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
| x                      | Specifies the horizontal position, in screen coordinates, of the pop-up menu. Depending on the value of the fuFlags parameter, the menu can be left-aligned, right-aligned, or centered relative to this position.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
| y                      | Specifies the vertical position, in screen coordinates, of the top of the menu on the screen.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
| nReserved              | Reserved; must be zero.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
| hwnd                   | Identifies the window that owns the pop-up menu. This window receives all <u>WM_COMMAND</u> messages from the menu. The window will not receive <u>WM_COMMAND</u> messages until TrackPopupMenu returns.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
| lprc                   | Points to a <u>RECT</u> structure that contains the screen coordinates of a rectangle in which the user can click without dismissing the pop-up menu. If this parameter is NULL, the pop-up menu is dismissed if the user clicks outside the pop-up menu.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |
| Returns                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                        |                                                                                               |                      |                                                                                                              |                       |                                                                                                               |       |         |                       |                                                        |                        |                                                                             |

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Example

The following example creates and tracks a pop-up menu when the user clicks the left mouse button:

```
POINT ptCurrent;  
HMENU hmenu;  
  
ptCurrent = MAKEPOINT(lParam);  
hmenu = CreatePopupMenu();  
AppendMenu(hmenu, MF_ENABLED, IDM_ELLIPSE, "Ellipse");  
AppendMenu(hmenu, MF_ENABLED, IDM_SQUARE, "Square");  
AppendMenu(hmenu, MF_ENABLED, IDM_TRIANGLE, "Triangle");  
ClientToScreen(hwnd, &ptCurrent);  
TrackPopupMenu(hmenu, TPM_LEFTALIGN, ptCurrent.x,  
    ptCurrent.y, 0, hwnd, NULL);
```

See Also

CreatePopupMenu, GetSubMenu, RECT

### Windows 3.1 changes

Applications receive WM\_COMMAND messages from the pop-up menu after the TrackPopupMenu function returns. For earlier versions of Windows, applications received WM\_COMMAND before TrackPopupMenu returned.

The seventh parameter to the TrackPopupMenu function is no longer reserved:

| Parameter   | Description                                                                                                                                                                                                                                                           |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>lprc</u> | Points to a <u>RECT</u> structure that contains the screen coordinates of a rectangle within which the user can click without dismissing the pop-up menu. If this parameter is <u>NULL</u> , the pop-up menu is dismissed if the user clicks outside the pop-up menu. |

The following constants have been added:

TPM\_CENTERALIGN  
TPM\_LEFTALIGN  
TPM\_RIGHTALIGN  
TPM\_RIGHTBUTTON

TPM\_CENTRALIGN 0x0004

Centers the pop-up menu horizontally relative to the coordinate specified by the x parameter.

TPM\_CENTERALIGN 0x0004

TPM\_LEFTALIGN 0x0000

Positions the pop-up menu so that its left side is aligned with the coordinate specified by the x parameter.

TPM\_LEFTALIGN 0x0000

TPM\_RIGHTALIGN 0x0008

Positions the pop-up menu so that its right side is aligned with the coordinate specified by the x parameter.



TPM\_RIGHTALIGN 0x0008

TPM\_LEFTBUTTON 0x0000

Causes the pop-up menu to track the left mouse button.

TPM\_LEFTBUTTON 0x0000

TPM\_RIGHTBUTTON 0x0002

Causes the pop-up menu to track the right mouse button instead of the left.

TPM\_RIGHTBUTTON 0x0002

## TranslateAccelerator (2.x)

```
int TranslateAccelerator(hwnd, haccl, lpmsg)
HWND hwnd;          /* handle of window */
HACCEL haccl;      /* handle of accelerator table */
MSG FAR* lpmsg;    /* address of structure with message information */
```

The TranslateAccelerator function processes accelerator keys for menu commands. The function translates WM\_KEYUP and WM\_KEYDOWN messages to WM\_COMMAND or WM\_SYSCOMMAND messages if there is an entry for the accelerator key in the application's accelerator table.

| Parameter | Description                                                                                                                                                                                     |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd      | Identifies the window whose messages are to be translated.                                                                                                                                      |
| haccl     | Identifies an accelerator table (loaded by using the <u>LoadAccelerators</u> function).                                                                                                         |
| lpmsg     | Points to an <u>MSG</u> structure retrieved by a call to the <u>GetMessage</u> or <u>PeekMessage</u> function. The structure contains message information from the application's message queue. |

### Returns

The return value is nonzero if the message is translated. Otherwise, it is zero.

### Comments

The high-order word of the LPARAM parameter of the WM\_COMMAND or WM\_SYSCOMMAND message contains the value 1, to differentiate the message from messages sent by menus or controls.

WM\_COMMAND or WM\_SYSCOMMAND messages are sent directly to the window, rather than being posted to the application queue. The TranslateAccelerator function does not return until the message is processed.

Accelerator keystrokes that are defined to select items from the System menu are translated into WM\_SYSCOMMAND messages; all other accelerator keystrokes are translated into WM\_COMMAND messages.

When TranslateAccelerator returns a nonzero value (meaning that the message is translated), the application should not process the message again by using the TranslateMessage function.

Keystrokes in accelerator tables need not correspond to menu items.

If the accelerator keystroke does correspond to a menu item, the application is sent WM\_INITMENU and WM\_INITMENUPOPUP messages, just as if the user were trying to display the menu. However, these messages are not sent if any of the following conditions are present:

- The window is disabled.
- The menu item is disabled.
- The accelerator keystroke does not correspond to an item on the System menu and the window is minimized.
- A mouse capture is in effect (for more information, see the description of the SetCapture function)

If the window is the active window and there is no keyboard focus (generally the case if the window is minimized), WM\_SYSKEYUP and WM\_SYSKEYDOWN messages are translated instead of WM\_KEYUP and WM\_KEYDOWN messages.

If an accelerator keystroke that corresponds to a menu item occurs when the window that owns the menu is minimized, no WM\_COMMAND message is sent. However, if an accelerator keystroke that does not match any of the items on the window's menu or the System menu occurs, a WM\_COMMAND message is sent, even if the window is minimized.

### See Also

GetMessage, LoadAccelerators, PeekMessage, SetCapture, MSG

### TranslateMDISysAccel (3.0)

```
BOOL TranslateMDISysAccel(hwndClient, lpmsg)
HWND hwndClient;    /* handle of parent MDI client window */
MSG FAR* lpmsg;    /*
address of structure with message data
/
```

The TranslateMDISysAccel function processes accelerator keystrokes for the given multiple document interface (MDI) child window. The function translates WM\_KEYUP and WM\_KEYDOWN messages to WM\_SYSCOMMAND messages.

| Parameter  | Description                                                                                                                                                                                     |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwndClient | Identifies the parent MDI client window.                                                                                                                                                        |
| lpmsg      | Points to an <u>MSG</u> structure retrieved by a call to the <u>GetMessage</u> or <u>PeekMessage</u> function. The structure contains message information from the application's message queue. |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

#### Comments

The high-order word of the lParam parameter of the WM\_SYSCOMMAND message contains the value 1, to differentiate the message from messages sent by menus or controls.

#### See Also

GetMessage, PeekMessage, MSG, WM\_SYSCOMMAND

## TranslateMessage (2.x)

```
BOOL TranslateMessage(lpmsg)
const MSG FAR* lpmsg;          /* address of MSG structure */
```

The TranslateMessage function translates virtual-key messages into character messages, as follows:

- WM\_KEYDOWN/WM\_KEYUP combinations produce a WM\_CHAR or WM\_DEADCHAR message.
- WM\_SYSKEYDOWN/WM\_SYSKEYUP combinations produce a WM\_SYSCHAR or WM\_SYSDEADCHAR message.

The character messages are posted to the application's message queue, to be read the next time the application calls the GetMessage or PeekMessage function.

| Parameter | Description                                                                                                                                                                                     |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpmsg     | Points to an <u>MSG</u> structure retrieved by a call to the <u>GetMessage</u> or <u>PeekMessage</u> function. The structure contains message information from the application's message queue. |

### Returns

The return value is nonzero if the message is WM\_KEYDOWN, WM\_KEYUP, WM\_SYSKEYDOWN, or WM\_SYSKEYUP, regardless of whether the key that was pressed or released generates a WM\_CHAR message. Otherwise, the return value is zero.

### Comments

The TranslateMessage function does not modify the message pointed to by the lpmsg parameter.

TranslateMessage produces WM\_CHAR messages only for keys that are mapped to ASCII characters by the keyboard driver.

An application should not call TranslateMessage if the application processes virtual-key messages for some other purpose. For instance, an application should not call TranslateMessage if the TranslateAccelerator function returns nonzero.

### See Also

GetMessage, PeekMessage, TranslateAccelerator



## TransmitCommChar (2.x)

```
int TransmitCommChar(idComDev, chTransmit)
int idComDev;          /* communications device */
char chTransmit;      /* character to transmit */
/
```

The TransmitCommChar function places the specified character at the head of the transmission queue for the specified device.

| Parameter  | Description                                                                                                     |
|------------|-----------------------------------------------------------------------------------------------------------------|
| idComDev   | Specifies the communications device to transmit the character. The <u>OpenComm</u> function returns this value. |
| chTransmit | Specifies the character to be transmitted.                                                                      |

### Returns

The return value is zero if the function is successful. It is less than zero if the character cannot be transmitted.

### Comments

The TransmitCommChar function cannot be called repeatedly if the device is not transmitting. Once TransmitCommChar places a character in the transmission queue, the character must be transmitted before the function can be called again. TransmitCommChar returns an error if the previous character has not yet been sent.

### Example

The following example uses the TransmitCommChar function to send characters from the keyboard to the communications port:

```
case WM_CHAR:
    ch = (char)wParam;
    TransmitCommChar(idComDev, ch);

    /* Add a linefeed for every carriage return. */

    if (ch == 0x0d)
        TransmitCommChar(idComDev, 0x0a);

    break;
```

### See Also

OpenComm, WriteComm

## UngetCommChar (2.x)

```
int UngetCommChar(idComDev, chUnget)
int idComDev;      /* communications device */
char chUnget;     /* character to place in queue */
/
```

The UngetCommChar function places the specified character back in the receiving queue. The next read operation will return this character first.

| Parameter | Description                                                                                                           |
|-----------|-----------------------------------------------------------------------------------------------------------------------|
| idComDev  | Specifies the communications device that will receive the character. The <u>OpenComm</u> function returns this value. |
| chUnget   | Specifies the character to be placed in the receiving queue.                                                          |

### Returns

The return value is zero if the function is successful. Otherwise, it is less than zero.

### Comments

Consecutive calls to the UngetCommChar function are not permitted. The character placed in the queue must be read before this function can be called again.

## UnhookWindowsHook (2.x)

BOOL UnhookWindowsHook(idHook, hkprc)

int idHook; /\* type of hook function to remove \*/

HOOKPROC hkprc;

\* hook function procedure-instance address

/

The UnhookWindowsHook function is obsolete but has been retained for backward compatibility with Windows versions 3.0 and earlier. Applications written for Windows version 3.1 should use the UnhookWindowsHookEx function.

The UnhookWindowsHook function removes an application-defined hook function from a chain of hook functions. A hook function processes events before they are sent to an application's message loop in the WinMain function.

| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|----------------|---------------------------------------------------------------------------------------------------------------------------|--------|------------------------------------------------------------------------------------------------------------------------------------|----------|------------------------------------------------------------------------------------------------------------------|---------------|-----------------------------------------------------------------------------------------------------------------|-------------|----------------------------------------------------------------------------------------------------------------------------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------|------------------|----------------------------------------------------------------------------------------------------------------------------------|-------------|--------------------------------------------------------------------------------------------------------------------|----------|----------------------------------------------------------------------------------------------------------------------|--------------|------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------|
| idHook             | Specifies the type of function to be removed. This parameter can be one of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
|                    | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>WH_CALLWNDPROC</td><td>Removes a window-procedure filter. For more information, see the description of the <u>CallWndProc</u> callback function.</td></tr><tr><td>WH_CBT</td><td>Removes a computer-based training (CBT) filter. For more information, see the description of the <u>CBTProc</u> callback function.</td></tr><tr><td>WH_DEBUG</td><td>Removes a debugging filter. For more information, see the description of the <u>DebugProc</u> callback function.</td></tr><tr><td>WH_GETMESSAGE</td><td>Removes a message filter. For more information, see the description of the <u>GetMsgProc</u> callback function.</td></tr><tr><td>WH_HARDWARE</td><td>Removes a nonstandard hardware-message filter. For more information, see the description of the <u>HardwareProc</u> callback function.</td></tr><tr><td>WH_JOURNALPLAYBACK</td><td>Removes a journaling playback filter. For more information, see the description of the <u>JournalPlaybackProc</u> callback function.</td></tr><tr><td>WH_JOURNALRECORD</td><td>Removes a journaling record filter. For more information, see the description of the <u>JournalRecordProc</u> callback function.</td></tr><tr><td>WH_KEYBOARD</td><td>Removes a keyboard filter. For more information, see the description of the <u>KeyboardProc</u> callback function.</td></tr><tr><td>WH_MOUSE</td><td>Removes a mouse-message filter. For more information, see the description of the <u>MouseProc</u> callback function.</td></tr><tr><td>WH_MSGFILTER</td><td>Removes a message filter. For more information, see the description of the <u>MessageProc</u> callback function.</td></tr><tr><td>WH_SYSMSGFILTER</td><td>Removes a system-wide message filter. For more information, see the description of the <u>SysMsgProc</u> callback function.</td></tr></tbody></table> | Value | Meaning | WH_CALLWNDPROC | Removes a window-procedure filter. For more information, see the description of the <u>CallWndProc</u> callback function. | WH_CBT | Removes a computer-based training (CBT) filter. For more information, see the description of the <u>CBTProc</u> callback function. | WH_DEBUG | Removes a debugging filter. For more information, see the description of the <u>DebugProc</u> callback function. | WH_GETMESSAGE | Removes a message filter. For more information, see the description of the <u>GetMsgProc</u> callback function. | WH_HARDWARE | Removes a nonstandard hardware-message filter. For more information, see the description of the <u>HardwareProc</u> callback function. | WH_JOURNALPLAYBACK | Removes a journaling playback filter. For more information, see the description of the <u>JournalPlaybackProc</u> callback function. | WH_JOURNALRECORD | Removes a journaling record filter. For more information, see the description of the <u>JournalRecordProc</u> callback function. | WH_KEYBOARD | Removes a keyboard filter. For more information, see the description of the <u>KeyboardProc</u> callback function. | WH_MOUSE | Removes a mouse-message filter. For more information, see the description of the <u>MouseProc</u> callback function. | WH_MSGFILTER | Removes a message filter. For more information, see the description of the <u>MessageProc</u> callback function. | WH_SYSMSGFILTER | Removes a system-wide message filter. For more information, see the description of the <u>SysMsgProc</u> callback function. |
| Value              | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
| WH_CALLWNDPROC     | Removes a window-procedure filter. For more information, see the description of the <u>CallWndProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
| WH_CBT             | Removes a computer-based training (CBT) filter. For more information, see the description of the <u>CBTProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
| WH_DEBUG           | Removes a debugging filter. For more information, see the description of the <u>DebugProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
| WH_GETMESSAGE      | Removes a message filter. For more information, see the description of the <u>GetMsgProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
| WH_HARDWARE        | Removes a nonstandard hardware-message filter. For more information, see the description of the <u>HardwareProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
| WH_JOURNALPLAYBACK | Removes a journaling playback filter. For more information, see the description of the <u>JournalPlaybackProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
| WH_JOURNALRECORD   | Removes a journaling record filter. For more information, see the description of the <u>JournalRecordProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
| WH_KEYBOARD        | Removes a keyboard filter. For more information, see the description of the <u>KeyboardProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
| WH_MOUSE           | Removes a mouse-message filter. For more information, see the description of the <u>MouseProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
| WH_MSGFILTER       | Removes a message filter. For more information, see the description of the <u>MessageProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
| WH_SYSMSGFILTER    | Removes a system-wide message filter. For more information, see the description of the <u>SysMsgProc</u> callback function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |
| hkprc              | Specifies the procedure-instance address of the application-defined filter function to remove.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                |                                                                                                                           |        |                                                                                                                                    |          |                                                                                                                  |               |                                                                                                                 |             |                                                                                                                                        |                    |                                                                                                                                      |                  |                                                                                                                                  |             |                                                                                                                    |          |                                                                                                                      |              |                                                                                                                  |                 |                                                                                                                             |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero.

### Comments

The `UnhookWindowsHook` function calls the hook chain, causing the hook function to receive a negative value for the `idHook` parameter. The hook function must then call the `DefHookProc` function, which removes the hook function from the chain.

See Also

`SetWindowsHook`

## Windows 3.1 changes

The following new hook types have been added:

| Value       | Meaning                                         |
|-------------|-------------------------------------------------|
| WH_CBT      | Removes a Computer-Based Training (CBT) filter. |
| WH_DEBUG    | Removes a debugging filter.                     |
| WH_HARDWARE | Removes a non-standard hardware-message filter. |
| WH_MOUSE    | Removes a mouse-message filter.                 |

## UnhookWindowsHookEx (3.1)

```
BOOL UnhookWindowsHookEx(hhook)
HHOOK hhook;      /* handle of hook function to remove */
```

The `UnhookWindowsHookEx` function removes an application-defined hook function from a chain of hook functions. A hook function processes events before they are sent to an application's message loop in the `WinMain` function.

| Parameter | Description                                                                                                                                       |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| hook      | Identifies the hook function to be removed. This is the value returned by the <code>SetWindowsHookEx</code> function when the hook was installed. |

### Returns

The return value is nonzero if the function is successful. It is zero if the hook cannot be found.

### Comments

The `UnhookWindowsHookEx` function must be used in combination with the `SetWindowsHookEx` function.

### Example

The following example uses the `UnhookWindowsHookEx` function to remove a message filter that was used to provide context-sensitive help for a dialog box:

```
DLGPROC lpfnAboutProc;
HOOKPROC lpfnFilterProc;
HHOOK hhook;

case IDM_ABOUT:
    lpfnAboutProc = (DLGPROC) MakeProcInstance(About, hinst);
    lpfnFilterProc = (HOOKPROC) MakeProcInstance(FilterFunc, hinst);
    hhook = SetWindowsHookEx(WH_MSGFILTER, lpfnFilterProc,
        hinst, (HTASK) NULL);

    DialogBox(hinst, "AboutBox", hwnd, lpfnAboutProc);

    UnhookWindowsHookEx(hhook);
    FreeProcInstance((FARPROC) lpfnFilterProc);
    FreeProcInstance((FARPROC) lpfnAboutProc);

    break;
```

### See Also

[CallNextHookEx](#), [SetWindowsHookEx](#)

## UnionRect (2.x)

```
BOOL UnionRect(lprcDst, lprcSrc1, lprcSrc2)
RECT FAR* lprcDst;          /* address of structure for union */
const RECT FAR* lprcSrc1;   /* address of structure with 1st rect.
/
const RECT FAR* lprcSrc2;   /* address of structure with 2nd rect.
/
```

The UnionRect function creates the union of two rectangles. The union is the smallest rectangle that contains both source rectangles.

| Parameter | Description                                                                                                                            |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------|
| lprcDst   | Points to a <u>RECT</u> structure to receive a rectangle containing the rectangles pointed to by the lprcSrc1 and lprcSrc2 parameters. |
| lprcSrc1  | Points to a <u>RECT</u> structure that contains the first source rectangle.                                                            |
| lprcSrc2  | Points to a <u>RECT</u> structure that contains the second source rectangle.                                                           |

### Returns

The return value is nonzero if the function is successful--that is, if the lprcDst parameter contains a nonempty rectangle. It is zero if the rectangle is empty or an error occurs.

### Comments

Windows ignores the dimensions of an empty rectangle--that is, a rectangle that has no height or no width.

### See Also

InflateRect, IntersectRect, OffsetRect, SubtractRect, RECT

## UnregisterClass (3.0)

```
BOOL UnregisterClass(lpszClassName, hinst)
LPCSTR lpszClassName;          /* address of class-name string */
HINSTANCE hinst;               /* handle of application instance */
/
```

The UnregisterClass function removes a window class, freeing the storage required for the class.

| Parameter     | Description                                                                                                                                                                                                                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszClassName | Points to a null-terminated string containing the class name. This class name must have been registered by a previous call to the <u>RegisterClass</u> function with a valid <u>hInstance</u> member of the <u>WNDCLASS</u> structure. Predefined classes, such as dialog box controls, cannot be unregistered. |
| hinst         | Identifies the instance of the module that created the class.                                                                                                                                                                                                                                                   |

### Returns

The return value is nonzero if the function successful. It is zero if the class could not be found or if a window exists that was created with the class.

### Comments

Before calling this function, an application should destroy all windows that were created with the specified class.

### See Also

RegisterClass, WNDCLASS



## UpdateWindow (2.x)

```
void UpdateWindow(hwnd)  
HWND hwnd;    /* handle of window */
```

The UpdateWindow function updates the client area of the given window by sending a WM\_PAINT message to the window if the update region for the window is not empty. The function sends a WM\_PAINT message directly to the window procedure of the given window, bypassing the application queue. If the update region is empty, no message is sent.

| Parameter | Description                          |
|-----------|--------------------------------------|
| hwnd      | Identifies the window to be updated. |

### Returns

This function does not return a value.

### See Also

ExcludeUpdateRgn, GetUpdateRect, GetUpdateRgn, InvalidateRect, InvalidateRgn, WM\_PAINT

## ValidateRect (2.x)

```
void ValidateRect(hwnd, lprc)
HWND hwnd;          /* handle of window */
const RECT FAR* lprc;
/* address of structure with validation rect.
/
```

The `ValidateRect` function validates the client area within the given rectangle by removing the rectangle from the update region of the given window.

| Parameter         | Description                                                                                                                                                                                               |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>hwnd</code> | Identifies the window whose update region is to be modified.                                                                                                                                              |
| <code>lprc</code> | Points to a <u>RECT</u> structure that contains the client coordinates of the rectangle to be removed from the update region. If this parameter is <code>NULL</code> , the entire client area is removed. |

### Returns

This function does not return a value.

### Comments

The `BeginPaint` function automatically validates the entire client area. Neither the `ValidateRect` nor the `ValidateRgn` function should be called if a portion of the update region needs to be validated before the next `WM_PAINT` message is generated.

Windows continues to generate `WM_PAINT` messages until the current update region is validated.

### See Also

`BeginPaint`, `InvalidateRect`, `InvalidateRgn`, `ValidateRgn`, `RECT`, `WM_PAINT`

## ValidateRgn (2.x)

```
void ValidateRgn(hwnd, hrgn)
HWND hwnd;    /* handle of window    */
HRGN hrgn;    /* handle of valid region */
```

The `ValidateRgn` function validates the client area within the given region by removing the region from the current update region of the specified window.

| Parameter         | Description                                                                                                                                                 |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>hwnd</code> | Identifies the window whose update region is to be modified.                                                                                                |
| <code>hrgn</code> | Identifies a region that defines the area to be removed from the update region. If this parameter is <code>NULL</code> , the entire client area is removed. |

### Returns

This function does not return a value.

### Comments

The given region must have been created by a region function. The region coordinates are assumed to be client coordinates.

The `BeginPaint` function automatically validates the entire client area. Neither the `ValidateRect` nor the `ValidateRgn` function should be called if a portion of the update region must be validated before the next `WM_PAINT` message is generated.

### See Also

[BeginPaint](#), [InvalidateRect](#), [InvalidateRgn](#), [ValidateRect](#), [WM\\_PAINT](#)

## WaitMessage (2.x)

void WaitMessage(void)

The WaitMessage function yields control to other applications when an application has no other tasks to perform. The WaitMessage function suspends the application and does not return until a new message is placed in the application's queue.

### Returns

This function does not return a value.

### Comments

The WaitMessage function normally returns immediately if there is a message in the queue. If an application has used the PeekMessage function but not removed the message, however, WaitMessage does not return until the message is received. Applications that use the PeekMessage function should remove any retrieved messages from the queue before calling WaitMessage.

The GetMessage, PeekMessage, and WaitMessage functions yield control to other applications. Using these functions is the only way to allow other applications to run. Applications that do not call any of these functions for long periods prevent other applications from running.

### See Also

GetMessage, PeekMessage

## WindowFromPoint (2.x)

```
HWND WindowFromPoint(pt)  
POINT pt;          /* structure with point */
```

The WindowFromPoint function retrieves the handle of the window that contains the specified point.

| Parameter | Description                                                                                        |
|-----------|----------------------------------------------------------------------------------------------------|
| pt        | Specifies a <u>POINT</u> structure that defines the screen coordinates of the point to be checked. |

### Returns

The return value is the handle of the window in which the point lies, if the function is successful. The return value is NULL if no window exists at the specified point.

### Comments

The WindowFromPoint function does not retrieve the handle of a hidden, disabled, or transparent window, even if the point is within the window. An application should use the ChildWindowFromPoint function for a nonrestrictive search.

### See Also

ChildWindowFromPoint

■ **WinHelp (3.0)**

```

BOOL WinHelp(hwnd, lpszHelpFile, fuCommand, dwData)
HWND hwnd; /* handle of window requesting help */
LPCSTR lpszHelpFile; /* address of directory-path string */
/
UINT fuCommand; /* type of help */
/
DWORD dwData; /* additional data */
/

```

The WinHelp function starts Windows Help (WINHELP.EXE) and passes optional data indicating the nature of the help requested by the application. The application specifies the name and, where required, the path of the help file that the Help application is to display.

| Parameter    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hwnd         | Identifies the window requesting Help. The WinHelp function uses this handle to keep track of which applications have requested Help.                                                                                                                                                                                                                                                                                                           |
| lpszHelpFile | Points to a null-terminated string containing the path, if necessary, and the name of the help file that the Help application is to display. The filename may be followed by an angle bracket (>) and the name of a secondary window if the topic is to be displayed in a secondary window rather than in the primary window. The name of the secondary window must have been defined in the [WINDOWS] section of the Help project (.HPJ) file. |
| fuCommand    | Specifies the type of help requested. For a list of possible values and how they affect the value to place in the dwData parameter, see the following Comments section.                                                                                                                                                                                                                                                                         |
| dwData       | Specifies additional data. The value used depends on the value of the fuCommand parameter. For a list of possible values, see the following Comments section.                                                                                                                                                                                                                                                                                   |

**Returns**

The return value is nonzero if the function is successful. Otherwise, it is zero.

**Comments**

Before closing the window that requested the help, the application must call WinHelp with fuCommand set to HELP\_QUIT. Until all applications have done this, Windows Help does not terminate.

The following table shows the possible values for the fuCommand parameter and the corresponding formats of the dwData parameter:

| fuCommand         | dwData                                                                                                                         | Action                                                                                                                                                                                            |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HELP_CONTEXT      | An unsigned long integer containing the context number for the topic.                                                          | Displays Help for a particular topic identified by a context number that has been defined in the [MAP] section of the .HPJ file.                                                                  |
| HELP_CONTENTS     | Ignored; applications should set to 0L                                                                                         | Displays the Help contents topic as defined by the Contents option in the [OPTIONS] section of the .HPJ file.                                                                                     |
| HELP_SETCONTENTS  | An unsigned long integer containing the context number for the topic the application wants to designate as the Contents topic. | Determines which Contents topic Help should display when a user presses the F1 key.                                                                                                               |
| HELP_CONTEXTPOPUP | An unsigned long integer containing the context number for a topic.                                                            | Displays in a pop-up window a particular Help topic identified by a context number that has been defined in the [MAP] section of the .HPJ file.                                                   |
| HELP_KEY          | A long pointer to a string that contains a keyword for the desired topic.                                                      | Displays the topic found in the keyword list that matches the keyword passed in the dwData parameter if there is one exact match. If there is more than one match, displays the Search dialog box |

|                 |                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 |                                                                                                                                                                                                      | with the topics listed in the Go To list box.                                                                                                                                                                                                                                                                                                                                                                                                                  |
| HELP_PARTIALKEY | A long pointer to a string that contains a keyword for the desired topic.                                                                                                                            | Displays the topic found in the keyword list that matches the keyword passed in the dwData parameter if there is one exact match. If there is more than one match, displays the Search dialog box with the topics found listed in the Go To list box. If there is no match, displays the Search dialog box. If you just want to bring up the Search dialog box without passing a keyword (the third result), you should use a long pointer to an empty string. |
| HELP_MULTIKEY   | A long pointer to the <u>MULTIKEYHELP</u> structure, as defined in <u>WINDOWS.H</u> . This structure specifies the table footnote character and the keyword.                                         | Displays the Help topic identified by a keyword in an alternate key word table.                                                                                                                                                                                                                                                                                                                                                                                |
| HELP_COMMAND    | A long pointer to a string that contains a Help macro to be executed.                                                                                                                                | Executes a Help macro.                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| HELP_SETWINPOS  | A long pointer to the <u>HELPWININF</u> structure, as defined in <u>WINDOWS.H</u> . This structure specifies the size and position of the primary Help window or a secondary window to be displayed. | Displays the Help window if it is minimized or in memory, and positions it according to the data passed.                                                                                                                                                                                                                                                                                                                                                       |
| HELP_FORCEFILE  | Ignored; applications should set to 0L.                                                                                                                                                              | Ensures that WinHelp is displaying the correct Help file. If the correct Help file is currently displayed, there is no action. If the incorrect Help file is displayed, WinHelp opens the correct file.                                                                                                                                                                                                                                                        |
| HELP_HELPONHELP | Ignored; applications should set to 0L.                                                                                                                                                              | Displays the Contents topic of the designated Using Help file.                                                                                                                                                                                                                                                                                                                                                                                                 |
| HELP_QUIT       | Ignored; applications should set to 0L.                                                                                                                                                              | Informs the Help application that Help is no longer needed. If no other applications have asked for Help, Windows closes the Help application.                                                                                                                                                                                                                                                                                                                 |

See Also  
[MULTIKEYHELP](#), [Creating Help Files](#)

Windows 3.1 changes

The following constants have been added:

HELP\_SETCONTENTS (used to be HELP\_SETINDEX)  
HELP\_FORCEFILE  
HELP\_PARTIALKEY  
HELP\_CONTENTS (used to be HELP\_INDEX)  
HELP\_POPUPID



### WNetAddConnection (3.1)

UINT WNetAddConnection(lpszNetPath, lpszPassword, lpszLocalName)

```
LPSTR lpszNetPath;          /* address of network device */
LPSTR lpszPassword;        /*
address of password
/
LPSTR lpszLocalName;
* address of local device
/
```

The WNetAddConnection function redirects the specified local device (either a disk drive or a printer port) to the given shared device or remote server.

| Parameter     | Description                                                                                                                                                                                                                                |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszNetPath   | Points to a null-terminated string specifying the shared device or remote server.                                                                                                                                                          |
| lpszPassword  | Points to a null-terminated string specifying the network password for the given device or server.                                                                                                                                         |
| lpszLocalName | Points to a null-terminated string specifying the local drive or device to be redirected. All lpszLocalName strings (such as LPT1) are case-independent. Only the drive names A through Z and the device names LPT1 through LPT3 are used. |

#### Returns

The return value is one of the following:

| Value                       | Meaning                                                      |
|-----------------------------|--------------------------------------------------------------|
| <u>WN_SUCCESS</u>           | The function was successful.                                 |
| <u>WN_NOT_SUPPORTED</u>     | The function was not supported.                              |
| <u>WN_OUT_OF_MEMORY</u>     | The system was out of memory.                                |
| <u>WN_NET_ERROR</u>         | An error occurred on the network.                            |
| <u>WN_BAD_POINTER</u>       | The pointer was invalid.                                     |
| <u>WN_BAD_NETNAME</u>       | The network resource name was invalid.                       |
| <u>WN_BAD_LOCALNAME</u>     | The local device name was invalid.                           |
| <u>WN_BAD_PASSWORD</u>      | The password was invalid.                                    |
| <u>WN_ACCESS_DENIED</u>     | A security violation occurred.                               |
| <u>WN_ALREADY_CONNECTED</u> | The local device was already connected to a remote resource. |

#### See Also

WNetCancelConnection, WNetGetConnection

WN\_SUCCESS 0x0000

The function was successful.

WN\_SUCCESS 0x0000

WN\_NOT\_SUPPORTED 0x0001

The function was not supported.

WN\_NOT\_SUPPORTED 0x0001

WN\_OUT\_OF\_MEMORY 0x000B

The system was out of memory.

WN\_OUT\_OF\_MEMORY 0x000B

WN\_NET\_ERROR 0x0002

An error occurred on the network.



WN\_NET\_ERROR 0x0002

WN\_BAD\_POINTER 0x0004

The pointer was invalid.

WN\_BAD\_POINTER 0x0004

WN\_BAD\_NETNAME 0x0032

The network resource name was invalid.

WN\_BAD\_NETNAME 0x0032

WN\_BAD\_LOCALNAME 0x0033

The local device name was invalid.

WN\_BAD\_LOCALNAME 0x0033

WN\_BAD\_PASSWORD 0x0006

The password was invalid.



WN\_BAD\_PASSWORD 0x0006

WN\_ACCESS\_DENIED 0x0007

A security violation occurred.

WN\_ACCESS\_DENIED 0x0007

WN\_ALREADY\_CONNECTED 0x0034

The local device was already connected to a remote resource.

WN\_ALREADY\_CONNECTED 0x0034

### WNetCancelConnection (3.1)

```
UINT WNetCancelConnection(lpszName, fForce)
LPSTR lpszName;          /* address of device or resource */
BOOL fForce;             /* forced closure */
flag
/
```

The WNetCancelConnection function cancels a network connection.

| Parameter | Description                                                                                                                                                                                                                                                                            |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszName  | Points to the name of the redirected local device (such as LPT1 or D:).                                                                                                                                                                                                                |
| fForce    | Specifies whether any open files or open print jobs on the device should be closed before the connection is canceled. If this parameter is FALSE and there are open files or jobs, the connection should not be canceled and the function should return the WN_OPEN_FILES error value. |

#### Returns

The return value is one of the following:

| Value            | Meaning                                                                                          |
|------------------|--------------------------------------------------------------------------------------------------|
| WN_SUCCESS       | The function was successful.                                                                     |
| WN_NOT_SUPPORTED | The function was not supported.                                                                  |
| WN_OUT_OF_MEMORY | The system was out of memory.                                                                    |
| WN_NET_ERROR     | An error occurred on the network.                                                                |
| WN_BAD_POINTER   | The pointer was invalid.                                                                         |
| WN_BAD_VALUE     | The lpszName parameter was not a valid local device or network name.                             |
| WN_NOT_CONNECTED | The lpszName parameter was not a redirected local device or currently accessed network resource. |
| WN_OPEN_FILES    | Files were open and the fForce parameter was FALSE. The connection was not canceled.             |

#### See Also

WNetAddConnection, WNetGetConnection

### WNetGetConnection (3.1)

```
UINT WNetGetConnection(lpszLocalName, lpszRemoteName, cbRemoteName)
LPSTR lpszLocalName;          /* address of local device name */
LPSTR lpszRemoteName;
/* address of remote device name
/
UINT FAR* cbRemoteName;
/* max. number of bytes in buffer
/
```

The WNetGetConnection function returns the name of the network resource associated with the specified redirected local device.

| Parameter      | Description                                                                                                                                                                                |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszLocalName  | Points to a null-terminated string specifying the name of the redirected local device.                                                                                                     |
| lpszRemoteName | Points to the buffer to receive the null-terminated name of the remote network resource.                                                                                                   |
| cbRemoteName   | Points to a variable specifying the maximum number of bytes the buffer pointed to by lpszRemoteName can hold. The function sets this variable to the number of bytes copied to the buffer. |

#### Returns

The return value is one of the following:

| Value            | Meaning                                                      |
|------------------|--------------------------------------------------------------|
| WN_SUCCESS       | The function was successful.                                 |
| WN_NOT_SUPPORTED | The function was not supported.                              |
| WN_OUT_OF_MEMORY | The system was out of memory.                                |
| WN_NET_ERROR     | An error occurred on the network.                            |
| WN_BAD_POINTER   | The pointer was invalid.                                     |
| WN_BAD_VALUE     | The szLocalName parameter was not a valid local device.      |
| WN_NOT_CONNECTED | The szLocalName parameter was not a redirected local device. |
| WN_MORE_DATA     | The buffer was too small.                                    |

#### See Also

WNetAddConnection, WNetCancelConnection

## WriteComm (2.x)

```
int WriteComm(idComDev, lpvBuf, cbWrite)
int idComDev;          /* identifier of comm. device */
const void FAR* lpvBuf; /* address of data buffer */
int cbWrite;          /* number of bytes to write */
```

The WriteComm function writes to the specified communications device.

| Parameter | Description                                                                                 |
|-----------|---------------------------------------------------------------------------------------------|
| idComDev  | Specifies the device to receive the bytes. The <u>OpenComm</u> function returns this value. |
| lpvBuf    | Points to the buffer that contains the bytes to be written.                                 |
| cbWrite   | Specifies the number of bytes to be written.                                                |

### Returns

The return value specifies the number of bytes written, if the function is successful. The return value is less than zero if an error occurs, making the absolute value of the return value the number of bytes written.

### Comments

To determine what caused an error, use the GetCommError function to retrieve the error value and status.

For serial ports, the WriteComm function deletes data in the transmission queue if there is not enough room in the queue for the additional bytes. Before calling WriteComm, applications should check the available space in the transmission queue by using the GetCommError function. Also, applications should use the OpenComm function to set the size of the transmission queue to an amount no smaller than the size of the largest expected output string.

### See Also

GetCommError, OpenComm, TransmitCommChar



### wvsprintf (3.0)

```
int wvsprintf(lpszOutput, lpszFormat, lpvArglist)
LPSTR lpszOutput;          /* address of output destination */
LPCSTR lpszFormat;        /* address of format string */
/
const void FAR* lpvArglist;
/* address of array of arguments */
/
```

The wvsprintf function formats and stores a series of characters and values in a buffer. The items pointed to by the argument list are converted according to the corresponding format specification in the format string.

| Parameter  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszOutput | Points to a null-terminated string to receive the string formatted as specified in the lpszFormat parameter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| lpszFormat | Points to a null-terminated string that contains the format-control string. In addition to the standard ASCII characters, a format specification for each argument appears in this string. For more information about the format specification, see the description of the wsprintf function.                                                                                                                                                                                                                                                                                           |
| lpvArglist | Points to an array of 16-bit values, each of which specifies an argument for the format-control string. The number, type, and interpretation of the arguments depend on the corresponding format-control character sequences specified in the lpszFormat parameter. Each character or 16-bit integer (%c, %d, %x, %i) requires one word in lpvArglist. Long integers (%ld, %li, %lx) require two words, the low-order word of the integer followed by the high-order word. A string (%s) requires two words, the offset followed by the segment (which together make up a far pointer). |

#### Returns

The return value is the number of bytes stored in the lpszOutput string, not counting the terminating null character, if the function is successful.

#### See Also

wsprintf

## User functions

|                                  |                                                  |
|----------------------------------|--------------------------------------------------|
| <u>AdjustWindowRect</u>          | Computes required size of a window rectangle     |
| <u>AdjustWindowRectEx</u>        | Computes required size of a window rectangle     |
| <u>AnsiLower</u>                 | Converts a string to lowercase                   |
| <u>AnsiLowerBuff</u>             | Converts a string buffer to lowercase            |
| <u>AnsiNext</u>                  | Moves to the next character in a string          |
| <u>AnsiPrev</u>                  | Moves to the previous character in a string      |
| <u>AnsiUpper</u>                 | Converts a string to uppercase                   |
| <u>AnsiUpperBuff</u>             | Converts a string buffer to uppercase            |
| <u>AnyPopup</u>                  | Indicates if pop-up or overlapped window exists  |
| <u>AppendMenu</u>                | Appends a new item to a menu                     |
| <u>ArrangeIconicWindows</u>      | Arranges minimized child windows                 |
| <u>BeginDeferWindowPos</u>       | Creates a window-positioning structure           |
| <u>BeginPaint</u>                | Prepares a window for painting                   |
| <u>BringWindowToTop</u>          | Uncovers an overlapped window                    |
| <u>BuildCommDCB</u>              | Translates a device-definition string to a DCB   |
| <u>CallMsgFilter</u>             | Passes a message to a message-filter function    |
| <u>CallNextHookEx</u>            | Passes hook information down the hook chain      |
| <u>CallWindowProc</u>            | Passes a message to a window procedure           |
| <u>ChangeClipboardChain</u>      | Removes a window from the clipboard-viewer chain |
| <u>ChangeMenu</u>                | Not used in Windows 3.1                          |
| <u>CheckDlgButton</u>            | Changes a check mark by a dialog box button      |
| <u>CheckMenuItem</u>             | Changes a check mark by a menu item              |
| <u>CheckRadioButton</u>          | Places a check mark by a radio button            |
| <u>ChildWindowFromPoint</u>      | Determines the child window containing a point   |
| <u>ClearCommBreak</u>            | Restores character transmission                  |
| <u>ClientToScreen</u>            | Converts a client point to screen coordinates    |
| <u>ClipCursor</u>                | Confines the cursor to a specified rectangle     |
| <u>CloseClipboard</u>            | Closes the clipboard                             |
| <u>CloseComm</u>                 | Closes a communications device                   |
| <u>CloseWindow</u>               | Minimizes a window                               |
| <u>CloseDriver</u>               | Closes an installable driver                     |
| <u>CopyCursor</u>                | Copies a cursor                                  |
| <u>CopyIcon</u>                  | Copies an icon                                   |
| <u>CopyRect</u>                  | Copies the dimensions of a rectangle             |
| <u>CountClipboardFormats</u>     | Returns the number of clipboard formats          |
| <u>CreateCaret</u>               | Creates a new shape for the system caret         |
| <u>CreateCursor</u>              | Creates a cursor with the specified dimensions   |
| <u>CreateDialog</u>              | Creates a modeless dialog box                    |
| <u>CreateDialogIndirect</u>      | Creates modeless dialog box from memory template |
| <u>CreateDialogIndirectParam</u> | Creates modeless dialog box from memory template |
| <u>CreateDialogParam</u>         | Creates a modeless dialog box                    |
| <u>CreateIcon</u>                | Creates an icon with the specified dimensions    |
| <u>CreateMenu</u>                | Creates a menu                                   |
| <u>CreatePopupMenu</u>           | Creates a pop-up menu                            |
| <u>CreateWindow</u>              | Creates a window                                 |
| <u>CreateWindowEx</u>            | Creates a window                                 |
| <u>DefDlgProc</u>                | Does default window message processing           |
| <u>DefDriverProc</u>             | Calls the default installable-driver procedure   |
| <u>DeferWindowPos</u>            | Updates a multiple window-positioning structure  |
| <u>DefFrameProc</u>              | Does default MDI frame window message processing |
| <u>DefHookProc</u>               | Calls the next function in a hook-function chain |
| <u>DefMDIChildProc</u>           | Does default MDI child window message processing |
| <u>DefWindowProc</u>             | Calls the default window procedure               |
| <u>DeleteMenu</u>                | Deletes an item from a menu                      |
| <u>DestroyCaret</u>              | Destroys the current caret                       |
| <u>DestroyCursor</u>             | Destroys a cursor                                |
| <u>DestroyIcon</u>               | Destroys an icon                                 |
| <u>DestroyMenu</u>               | Destroys a menu                                  |
| <u>DestroyWindow</u>             | Destroys a window                                |
| <u>DialogBox</u>                 | Creates a modal dialog box                       |
| <u>DialogBoxIndirect</u>         | Creates modal dialog box from memory template    |
| <u>DialogBoxIndirectParam</u>    | Creates modal dialog box from memory template    |
| <u>DialogBoxParam</u>            | Creates a modal dialog box                       |

|                               |                                                  |
|-------------------------------|--------------------------------------------------|
| <u>DispatchMessage</u>        | Dispatches a message to a window                 |
| <u>DlgDirList</u>             | Fills a directory list box                       |
| <u>DlgDirListComboBox</u>     | Fills a directory list box                       |
| <u>DlgDirSelect</u>           | Retrieves a selection from a directory list box  |
| <u>DlgDirSelectEx</u>         | Retrieves a selection from a directory list box  |
| <u>DlgDirSelectComboBox</u>   | Retrieves a selection from a directory list box  |
| <u>DlgDirSelectComboBoxEx</u> | Retrieves a selection from a directory list box  |
| <u>DrawFocusRect</u>          | Draws a rectangle in the focus style             |
| <u>DrawIcon</u>               | Draws an icon in the specified device context    |
| <u>DrawMenuBar</u>            | Redraws the menu bar                             |
| <u>DrawText</u>               | Draws the formatted text in a rectangle          |
| <u>EmptyClipboard</u>         | Empties the clipboard and frees the data handles |
| <u>EnableCommNotification</u> | Enables or disables <u>WM_COMMNOTIFY</u> posting |
| <u>EnableHardwareInput</u>    | Controls mouse and keyboard input queuing        |
| <u>EnableMenuItem</u>         | Enables, disables, or grays a menu item          |
| <u>EnableScrollBar</u>        | Enables or disables scroll-bar arrows            |
| <u>EnableWindow</u>           | Sets the window-enable state                     |
| <u>EndDeferWindowPos</u>      | Updates position and size of multiple windows    |
| <u>EndDialog</u>              | Hides a modal dialog box                         |
| <u>EndPaint</u>               | Marks the end of painting in specified window    |
| <u>EnumChildWindows</u>       | Passes child-window handles to a callback        |
| <u>EnumClipboardFormats</u>   | Returns available clipboard formats              |
| <u>EnumProps</u>              | Passes property-list entries to a callback       |
| <u>EnumTaskWindows</u>        | Passes task's window handles to a callback       |
| <u>EnumWindows</u>            | Passes parent-window handles to a callback       |
| <u>EqualRect</u>              | Determines whether two rectangles are equal      |
| <u>EscapeCommFunction</u>     | Passes an extended function to a device          |
| <u>ExcludeUpdateRgn</u>       | Excludes updated region from clipping region     |
| <u>ExitWindows</u>            | Restarts or terminates Windows                   |
| <u>ExitWindowsExec</u>        | Terminates Windows and runs MS-DOS application   |
| <u>FillRect</u>               | Fills a rectangle with the specified brush       |
| <u>FindWindow</u>             | Returns window handle for class and window name  |
| <u>FlashWindow</u>            | Flashes a window once                            |
| <u>FlushComm</u>              | Flushes a transmit or receiving queue            |
| <u>FrameRect</u>              | Draws a window border with a specified brush     |
| <u>GetActiveWindow</u>        | Retrieves the handle of the active window        |
| <u>GetAsyncKeyState</u>       | Determines the key state                         |
| <u>GetCapture</u>             | Returns the handle for the mouse-capture window  |
| <u>GetCaretBlinkTime</u>      | Returns the caret blink rate                     |
| <u>GetCaretPos</u>            | Returns the current caret position               |
| <u>GetClassInfo</u>           | Returns window-class information                 |
| <u>GetClassLong</u>           | Returns a window-class long value                |
| <u>GetClassName</u>           | Returns a window-class name                      |
| <u>GetClassWord</u>           | Returns a window-class word value                |
| <u>GetClientRect</u>          | Returns client area coordinates of window        |
| <u>GetClipboardData</u>       | Returns a handle to the current clipboard data   |
| <u>GetClipboardFormatName</u> | Returns the registered clipboard format name     |
| <u>GetClipboardOwner</u>      | Returns the clipboard owner window handle        |
| <u>GetClipboardViewer</u>     | Returns first clipboard-viewer window handle     |
| <u>GetClipCursor</u>          | Returns cursor-confining rectangle coordinates   |
| <u>GetCommError</u>           | Returns the communications-device status         |
| <u>GetCommEventMask</u>       | Retrieves the device event mask                  |
| <u>GetCommState</u>           | Reads the communications-device status           |
| <u>GetCurrentTime</u>         | Returns the elapsed time since Windows started   |
| <u>GetCursor</u>              | Returns the current cursor handle                |
| <u>GetCursorPos</u>           | Returns the current cursor position              |
| <u>GetDC</u>                  | Returns the window device-context handle         |
| <u>GetDCEX</u>                | Retrieves the device-context handle              |
| <u>GetDesktopWindow</u>       | Returns desktop window handle                    |
| <u>GetDialogBaseUnits</u>     | Returns the dialog box base units                |
| <u>GetDlgCtrlID</u>           | Returns the handle of a child window             |
| <u>GetDlgItem</u>             | Returns dialog box control handle                |
| <u>GetDlgItemInt</u>          | Translates dialog box text into an integer       |
| <u>GetDlgItemText</u>         | Retrieves dialog box control text                |

|                                   |                                                    |
|-----------------------------------|----------------------------------------------------|
| <u>GetDoubleClickTime</u>         | Returns mouse double-click time                    |
| <u>GetDriverModuleHandle</u>      | Retrieves an installable-driver instance handle    |
| <u>GetDriverInfo</u>              | Retrieves installable-driver data                  |
| <u>GetFocus</u>                   | Returns the current focus window handle            |
| <u>GetFreeSystemResources</u>     | Returns percentage of free system-resource space   |
| <u>GetInputState</u>              | Returns mouse and keyboard status                  |
| <u>GetKeyboardState</u>           | Returns the status of virtual-keyboard keys        |
| <u>GetKeyState</u>                | Returns the specified virtual-key state            |
| <u>GetLastActivePopup</u>         | Determines most recently active pop-up window      |
| <u>GetMenu</u>                    | Returns the menu handle for a specified window     |
| <u>GetMenuCheckMarkDimensions</u> | Returns the default check mark bitmap dimensions   |
| <u>GetMenuItemCount</u>           | Returns the number of items in a menu              |
| <u>GetMenuItemID</u>              | Returns a menu-item identifier                     |
| <u>GetMenuState</u>               | Returns status flags for the specified menu item   |
| <u>GetMenuString</u>              | Copies a menu-item label into a buffer             |
| <u>GetMessage</u>                 | Retrieves a message from the message queue         |
| <u>GetMessageExtraInfo</u>        | Retrieves information about a hardware message     |
| <u>GetMessagePos</u>              | Returns the cursor position for the last message   |
| <u>GetMessageTime</u>             | Returns the time for the last message              |
| <u>GetNextDlgGroupItem</u>        | Returns handle of next or previous group control   |
| <u>GetNextDlgTabItem</u>          | Returns next or previous <u>WS_TABSTOP</u> control |
| <u>GetNextDriver</u>              | Enumerates installable-driver instances            |
| <u>GetNextWindow</u>              | Returns next or previous window manager window     |
| <u>GetOpenClipboardWindow</u>     | Returns handle to window that opened clipboard     |
| <u>GetParent</u>                  | Returns the parent window handle                   |
| <u>GetPriorityClipboardFormat</u> | Returns the first clipboard format                 |
| <u>GetProp</u>                    | Returns data handle from window property list      |
| <u>GetQueueStatus</u>             | Determines the queued message type                 |
| <u>GetScrollPos</u>               | Returns the current scroll-bar thumb position      |
| <u>GetScrollRange</u>             | Returns minimum and maximum scroll-bar positions   |
| <u>GetSubMenu</u>                 | Returns the pop-up menu handle                     |
| <u>GetSysColor</u>                | Returns the display-element color                  |
| <u>GetSysModalWindow</u>          | Returns the system-modal window handle             |
| <u>GetSystemDebugState</u>        | Returns system-state information to a debugger     |
| <u>GetSystemMenu</u>              | Provides access to the System menu                 |
| <u>GetSystemMetrics</u>           | Retrieves the system metrics                       |
| <u>GetTabbedTextExtent</u>        | Determines the dimensions of a tabbed string       |
| <u>GetTickCount</u>               | Returns amount of time Windows has been running    |
| <u>GetTimerResolution</u>         | Retrieves the timer resolution                     |
| <u>GetTopWindow</u>               | Returns handle for top child of given window       |
| <u>GetUpdateRect</u>              | Returns the window update region dimensions        |
| <u>GetUpdateRgn</u>               | Returns the window update region                   |
| <u>GetWindow</u>                  | Returns the specified window handle                |
| <u>GetWindowDC</u>                | Returns the window device context                  |
| <u>GetWindowLong</u>              | Returns long value from extra window memory        |
| <u>GetWindowPlacement</u>         | Returns window show state and min/max position     |
| <u>GetWindowRect</u>              | Retrieves window screen coordinates                |
| <u>GetWindowTask</u>              | Returns the task associated with a window          |
| <u>GetWindowText</u>              | Copies the window title-bar text to a buffer       |
| <u>GetWindowTextLength</u>        | Returns the length of window title bar text        |
| <u>GetWindowWord</u>              | Returns a word value from extra window memory      |
| <u>GlobalAddAtom</u>              | Adds a string to the system atom table             |
| <u>GlobalDeleteAtom</u>           | Decrements the reference count of a global atom    |
| <u>GlobalFindAtom</u>             | Retrieves string atom from global atom table       |
| <u>GlobalGetAtomName</u>          | Retrieves a global atom string                     |
| <u>GrayString</u>                 | Draws gray text at the specified location          |
| <u>hardware_event</u>             | Places a hardware message in the system queue      |
| <u>HideCaret</u>                  | Removes the caret from the screen                  |
| <u>HiliteMenuItem</u>             | Changes the highlight of a top-level menu item     |
| <u>InflateRect</u>                | Changes the rectangle dimensions                   |
| <u>InSendMessage</u>              | Determines if a window is processing SendMessage   |
| <u>InsertMenu</u>                 | Inserts a new item in a menu                       |
| <u>IntersectRect</u>              | Calculates a rectangle intersection                |
| <u>InvalidateRect</u>             | Adds a rectangle to the update region              |

|                                   |                                                  |
|-----------------------------------|--------------------------------------------------|
| <u>InvalidateRgn</u>              | Adds a region to the update region               |
| <u>InvertRect</u>                 | Inverts a rectangular region                     |
| <u>IsCharAlpha</u>                | Determines if a character is alphabetic          |
| <u>IsCharAlphaNumeric</u>         | Determines if a character is alphanumeric        |
| <u>IsCharLower</u>                | Determines if a character is lowercase           |
| <u>IsCharUpper</u>                | Determines if a character is uppercase           |
| <u>IsChild</u>                    | Determines if a window is a child window         |
| <u>IsClipboardFormatAvailable</u> | Determines availability of data in given format  |
| <u>IsDialogMessage</u>            | Determines if a message is for a dialog box      |
| <u>IsDlgButtonChecked</u>         | Determines the state of a button control         |
| <u>IsIconic</u>                   | Determines if a window is minimized              |
| <u>IsMenu</u>                     | Determines if a menu handle is valid             |
| <u>IsRectEmpty</u>                | Determines whether a rectangle is empty          |
| <u>IsWindow</u>                   | Determines if a window handle is valid           |
| <u>IsWindowEnabled</u>            | Determines if a window accepts user input        |
| <u>IsWindowVisible</u>            | Determines the visibility state of a window      |
| <u>IsZoomed</u>                   | Determines if a window is maximized              |
| <u>KillTimer</u>                  | Removes a timer                                  |
| <u>LoadAccelerators</u>           | Loads an accelerator table                       |
| <u>LoadBitmap</u>                 | Loads a bitmap resource                          |
| <u>LoadCursor</u>                 | Loads a cursor resource                          |
| <u>LoadIcon</u>                   | Loads an icon resource                           |
| <u>LoadMenu</u>                   | Loads a menu resource                            |
| <u>LoadMenuIndirect</u>           | Obtains a menu handle for a menu template        |
| <u>LoadString</u>                 | Loads a string resource                          |
| <u>LockInput</u>                  | Locks input to all tasks except the current one  |
| <u>LockWindowUpdate</u>           | Disables or reenables drawing in a window        |
| <u>lstrcmp</u>                    | Compares two character strings                   |
| <u>lstrcmpi</u>                   | Compares two character strings                   |
| <u>MapDialogRect</u>              | Maps dialog box units to pixels                  |
| <u>MessageBeep</u>                | Generates a beep                                 |
| <u>MessageBox</u>                 | Creates a message box window                     |
| <u>MapWindowPoints</u>            | Converts points to another coordinate system     |
| <u>ModifyMenu</u>                 | Changes an existing menu item                    |
| <u>MoveWindow</u>                 | Changes the position and dimensions of a window  |
| <u>OffsetRect</u>                 | Moves a rectangle by an offset                   |
| <u>OpenClipboard</u>              | Opens the clipboard                              |
| <u>OpenComm</u>                   | Opens a communications device                    |
| <u>OpenDriver</u>                 | Opens an installable driver                      |
| <u>OpenIcon</u>                   | Activates a minimized window                     |
| <u>PeekMessage</u>                | Checks the message queue                         |
| <u>PostAppMessage</u>             | Posts a message to an application                |
| <u>PostMessage</u>                | Places a message in a window message queue       |
| <u>PostQuitMessage</u>            | Tells Windows that an application is terminating |
| <u>PtInRect</u>                   | Determines if a point is in a rectangle          |
| <u>QuerySendMessage</u>           | Determines if a message originated within a task |
| <u>ReadComm</u>                   | Reads from a communications device               |
| <u>RealizePalette</u>             | Maps entries from logical to system palette      |
| <u>RedrawWindow</u>               | Updates a client rectangle or region             |
| <u>RegisterClass</u>              | Registers a window class                         |
| <u>RegisterClipboardFormat</u>    | Registers a new clipboard format                 |
| <u>RegisterWindowMessage</u>      | Defines a new and unique window message          |
| <u>ReleaseCapture</u>             | Releases mouse capture                           |
| <u>ReleaseDC</u>                  | Frees a device context                           |
| <u>RemoveMenu</u>                 | Deletes a menu item and pop-up menu              |
| <u>RemoveProp</u>                 | Removes a property-list entry                    |
| <u>ReplyMessage</u>               | Replies to SendMessage                           |
| <u>ScreenToClient</u>             | Converts a screen point to client coordinates    |
| <u>ScrollDC</u>                   | Scrolls a rectangle horizontally and vertically  |
| <u>ScrollWindow</u>               | Scrolls a window client area                     |
| <u>ScrollWindowEx</u>             | Scrolls a window client area                     |
| <u>SelectPalette</u>              | Selects a palette into a device context          |
| <u>SendDlgItemMessage</u>         | Sends a message to a dialog box control          |
| <u>SendDriverMessage</u>          | Sends a message to an installable driver         |

|                             |                                                  |
|-----------------------------|--------------------------------------------------|
| <u>SendMessage</u>          | Sends a message to a window                      |
| <u>SetActiveWindow</u>      | Makes a top-level window active                  |
| <u>SetCapture</u>           | Sets the mouse capture to a window               |
| <u>SetCaretBlinkTime</u>    | Sets the caret blink rate                        |
| <u>SetCaretPos</u>          | Sets the caret position                          |
| <u>SetClassLong</u>         | Sets a long value in extra class memory          |
| <u>SetClassWord</u>         | Sets a word value in extra class memory          |
| <u>SetClipboardData</u>     | Sets the data in the clipboard                   |
| <u>SetClipboardViewer</u>   | Adds a window to the clipboard-viewer chain      |
| <u>SetCommBreak</u>         | Suspends character transmission                  |
| <u>SetCommEventMask</u>     | Enables events in a device event mask            |
| <u>SetCommState</u>         | Sets the communications-device state             |
| <u>SetCursor</u>            | Changes the mouse cursor                         |
| <u>SetCursorPos</u>         | Sets mouse-cursor position in screen coordinates |
| <u>SetDlgItemInt</u>        | Converts an integer to a dialog box text string  |
| <u>SetDlgItemText</u>       | Sets dialog box title or item text               |
| <u>SetDoubleClickTime</u>   | Sets the mouse double-click time                 |
| <u>SetFocus</u>             | Sets the input focus to a window                 |
| <u>SetKeyboardState</u>     | Sets the keyboard state table                    |
| <u>SetMenu</u>              | Sets the menu for a window                       |
| <u>SetMenuItemBitmaps</u>   | Associates bitmaps with a menu item              |
| <u>SetMessageQueue</u>      | Creates a new message queue                      |
| <u>SetParent</u>            | Changes a child's parent window                  |
| <u>SetProp</u>              | Adds or changes a property-list entry            |
| <u>SetRect</u>              | Sets rectangle dimensions                        |
| <u>SetRectEmpty</u>         | Creates an empty rectangle                       |
| <u>SetScrollPos</u>         | Sets scroll-bar thumb position                   |
| <u>SetScrollRange</u>       | Sets minimum and maximum scroll-bar positions    |
| <u>SetSysColors</u>         | Sets one or more system colors                   |
| <u>SetSysModalWindow</u>    | Makes a window the system-modal window           |
| <u>SetTimer</u>             | Installs a system timer                          |
| <u>SetWindowLong</u>        | Sets a long value in extra window memory         |
| <u>SetWindowPlacement</u>   | Sets window show state and min/max position      |
| <u>SetWindowPos</u>         | Sets window size, position, and order            |
| <u>SetWindowsHook</u>       | Installs a hook function                         |
| <u>SetWindowsHookEx</u>     | Installs a hook function                         |
| <u>SetWindowText</u>        | Sets text in a caption title or control window   |
| <u>SetWindowWord</u>        | Sets a word value in extra window memory         |
| <u>ShowCaret</u>            | Shows (unhides) the caret                        |
| <u>ShowCursor</u>           | Shows or hides the mouse cursor                  |
| <u>ShowOwnedPopups</u>      | Shows or hides pop-up windows                    |
| <u>ShowScrollBar</u>        | Shows or hides a scroll bar                      |
| <u>ShowWindow</u>           | Sets the window visibility state                 |
| <u>SubtractRect</u>         | Creates rect from difference of two rects        |
| <u>SwapMouseButton</u>      | Reverses the meaning of the mouse buttons        |
| <u>SystemParametersInfo</u> | Queries or sets system-wide parameters           |
| <u>TabbedTextOut</u>        | Writes a tabbed character string                 |
| <u>TrackPopupMenu</u>       | Displays and tracks a pop-up menu                |
| <u>TranslateAccelerator</u> | Processes menu command keyboard accelerators     |
| <u>TranslateMDISysAccel</u> | Processes MDI keyboard accelerators              |
| <u>TranslateMessage</u>     | Translates virtual-key messages                  |
| <u>TransmitCommChar</u>     | Places a character in the transmission queue     |
| <u>UngetCommChar</u>        | Puts character back in receiving queue           |
| <u>UnhookWindowsHook</u>    | Removes a filter function                        |
| <u>UnhookWindowsHookEx</u>  | Removes a function from the hook chain           |
| <u>UnionRect</u>            | Creates the union of two rectangles              |
| <u>UnregisterClass</u>      | Removes a window class                           |
| <u>UpdateWindow</u>         | Updates a window client area                     |
| <u>ValidateRect</u>         | Removes a rectangle from the update region       |
| <u>ValidateRgn</u>          | Removes a region from the update region          |
| <u>WaitMessage</u>          | Suspends an application and yields control       |
| <u>WindowFromPoint</u>      | Returns handle of window containing point        |
| <u>WinHelp</u>              | Invokes Windows Help                             |
| <u>WNetAddConnection</u>    | Adds network connections                         |

WNetCancelConnection  
WNetGetConnection  
WriteComm  
wvsprintf

Removes network connections  
Lists network connections  
Writes to a communications device  
Formats a string

## GetFileResource (3.1)

```
#include ver.h

BOOL GetFileResource(lpszFileName, lpszResType, lpszResID, dwFileOffset, dwResLen, lpvData)
LPCSTR lpszFileName; /* address of buffer for filename */
LPCSTR lpszResType; /* address of buffer for resource type */
/* address of buffer for resource ID */
LPCSTR lpszResID;
/* address of buffer for resource ID */
DWORD dwFileOffset;
/* resource offset in file */
DWORD dwResLen;
/* size of resource buffer */
void FAR* lpvData;
/* address of buffer for resource copy */
```

The GetFileResource function copies the specified resource from the specified file into the specified buffer. To obtain the appropriate buffer size, the application can call the [GetFileResourceSize](#) function before calling GetFileResource.

| Parameter    | Description                                                                                                                                                                                                                                                                    |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszFileName | Points to the buffer that contains the name of the file containing the resource.                                                                                                                                                                                               |
| lpszResType  | Points to a value that is created by using the <a href="#">MAKEINTRESOURCE</a> macro with the numbered resource type. This value is typically <a href="#">VS_FILE_INFO</a> .                                                                                                   |
| lpszResID    | Points to a value that is created by using the <a href="#">MAKEINTRESOURCE</a> macro with the numbered resource identifier. This value is typically <a href="#">VS_VERSION_INFO</a> .                                                                                          |
| dwFileOffset | Specifies the offset of the resource within the file. The <a href="#">GetFileResourceSize</a> function returns this value. If this parameter is NULL, the <a href="#">GetFileResource</a> function searches the file for the resource.                                         |
| dwResLen     | Specifies the buffer size, in bytes, identified by the lpvData parameter. The <a href="#">GetFileResourceSize</a> function returns the buffer size required to hold the resource. If the buffer is not large enough, the resource data is truncated to the size of the buffer. |
| lpvData      | Points to the buffer that will receive a copy of the resource. If the buffer is not large enough, the resource data is truncated.                                                                                                                                              |

### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero, indicating the function could not find the file, could not find the resource, or produced an MS-DOS error. The GetFileResource function returns no information about the type of error that occurred.

### Comments

If the dwFileOffset parameter is zero, the GetFileResource function determines the location of the resource by using the lpszResType and lpszResID parameters.

If dwFileOffset is not zero, GetFileResource assumes that dwFileOffset is the return value of [GetFileResourceSize](#) and, therefore, ignores lpszResType and lpszResID.

### See Also

[GetFileResourceSize](#)



### GetFileResourceSize (3.1)

```
#include ver.h
```

```
DWORD GetFileResourceSize(lpszFileName, lpszResType, lpszResID, lpdwFileOffset)
```

```
LPCSTR lpszFileName; /* address of buffer for filename */
```

```
LPCSTR lpszResType; /
```

```
* address of buffer for resource type
```

```
/
```

```
LPCSTR lpszResID;
```

```
* address of buffer for resource ID
```

```
/
```

```
DWORD FAR *lpdwFileOffset;
```

```
* address of resource offset in file
```

```
/
```

The GetFileResourceSize function searches the specified file for the resource of the specified type and identifier.

| Parameter      | Description                                                                                                                                                         |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszFileName   | Points to the buffer that contains the name of the file in which to search for the resource.                                                                        |
| lpszResType    | Points to a value that is created by using the <u>MAKEINTRESOURCE</u> macro with the numbered resource type. This value is typically <u>VS_FILE_INFO</u> .          |
| lpszResID      | Points to a value that is created by using the <u>MAKEINTRESOURCE</u> macro with the numbered resource identifier. This value is typically <u>VS_VERSION_INFO</u> . |
| lpdwFileOffset | Points to a 16-bit value that the GetFileResourceSize function fills with the offset to the resource within the file.                                               |

#### Returns

The return value is the size of the resource, in bytes. The return value is NULL if the function could not find the file, the file does not have any resources attached, or the function produced an MS-DOS error. The GetFileResourceSize function returns no information about the type of error that occurred.

#### See Also

GetFileResource

### GetFileVersionInfo (3.1)

```
#include ver.h

BOOL GetFileVersionInfo(lpszFileName, handle, cbBuf, lpvData)
LPCSTR lpszFileName;          /* address of buffer for filename */
DWORD handle;                /* file-
version information
/
DWORD cbBuf;
* size of buffer
/
void FAR* lpvData;
* address of buffer for file-version info
/
```

The GetFileVersionInfo function returns version information about the specified file. The application must call the GetFileVersionInfoSize function before calling GetFileVersionInfo to obtain the appropriate handle if the handle is not NULL.

| Parameter    | Description                                                                                                                                                                                                                                                                                         |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszFileName | Points to the buffer that contains the name of the file.                                                                                                                                                                                                                                            |
| handle       | Identifies the file-version information. The <u>GetFileVersionInfoSize</u> function returns this handle, or it may be NULL. If the handle parameter is NULL, the <u>GetFileVersionInfo</u> function searches the file for the version information.                                                  |
| cbBuf        | Specifies the buffer size, in bytes, identified by the lpvData parameter. The <u>GetFileVersionInfoSize</u> function returns the buffer size required to hold the file-version information. If the buffer is not large enough, the file-version information is truncated to the size of the buffer. |
| lpvData      | Points to the buffer that will receive the file-version information. This parameter is used by a subsequent call to the <u>VerQueryValue</u> function.                                                                                                                                              |

#### Returns

The return value is nonzero if the function is successful. Otherwise, it is zero, indicating the file does not exist or the handle parameter is invalid. The GetFileVersionInfo function returns no information about the type of error that occurred.

#### Comments

The file version information is organized in a VERSIONINFO statement.

Currently, the GetFileVersionInfo function recognizes only version-information created by Microsoft Resource Compiler (RC).

#### See Also

GetFileVersionInfoSize, VerQueryValue, VERSIONINFO

## GetFileVersionInfoSize (3.1)

```
#include ver.h
```

```
DWORD GetFileVersionInfoSize(lpszFileName, lpdwHandle)
LPCSTR lpszFileName;          /* address of buffer for filename */
DWORD FAR *lpdwHandle;       /* address of handle for info */
/
```

The `GetFileVersionInfoSize` function determines whether it can obtain version information from the specified file. If version information is available, `GetFileVersionInfoSize` returns the size of the buffer required to hold the version information. It also returns a handle that can be used in a subsequent call to the `GetFileVersionInfo` function.

| Parameter                 | Description                                                                                                                                                                                             |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>lpszFileName</code> | Points to the buffer that contains the name of the file.                                                                                                                                                |
| <code>lpdwHandle</code>   | Points to a 32-bit value that the <code>GetFileVersionInfoSize</code> function fills with the handle to the file-version information. The <code>GetFileVersionInfo</code> function can use this handle. |

### Returns

The return value is the buffer size, in bytes, required to hold the version information if the function is successful. The return value is `NULL` if the function could not find the file, could not find the version information, or produced an MS-DOS error. The `GetFileVersionInfoSize` function returns no information about the type of error that occurred.

### Comments

The file version information is organized in a `VERSIONINFO` statement.

### See Also

`GetFileVersionInfo`, `VERSIONINFO`

## GetSystemDir (3.1)

```
#include ver.h
```

```
UINT GetSystemDir(lpszWinDir, lpszBuf, cbBuf)
```

```
LPCSTR lpszWinDir; /* address of Windows directory */
```

```
LPSTR lpszBuf; /* address of
```

```
buffer for path
```

```
/
```

```
int cbBuf;
```

```
* size of buffer
```

```
/
```

The `GetSystemDir` function retrieves the path of the Windows system directory. This directory contains such files as Windows libraries, drivers, and fonts.

`GetSystemDir` is used by MS-DOS applications that set up Windows applications; it exists only in the static-link version of the File Installation library. Windows applications should use the `GetSystemDirectory` function to determine the Windows directory.

| Parameter               | Description                                                                                                     |
|-------------------------|-----------------------------------------------------------------------------------------------------------------|
| <code>lpszWinDir</code> | Points to the Windows directory retrieved by a previous call to the <u><code>GetWindowsDir</code></u> function. |
| <code>lpszBuf</code>    | Points to the buffer that is to receive the null-terminated string containing the path.                         |
| <code>cbBuf</code>      | Specifies the size, in bytes, of the buffer pointed to by the <code>lpszBuf</code> parameter.                   |

### Returns

The return value is the length of the string copied to the buffer, in bytes, including the terminating null character, if the function is successful. If the return value is greater than the `cbBuf` parameter, the return value is the size of the buffer required to hold the path. The return value is zero if the function fails.

### Comments

An application must call the `GetWindowsDir` function before calling the `GetSystemDir` function to obtain the correct `lpszWinDir` value.

The path that this function retrieves does not end with a backslash unless the Windows system directory is the root directory. For example, if the system directory is named `WINDOWS\SYSTEM` on drive C, the path of the system directory retrieved by this function is `C:\WINDOWS\SYSTEM`.

### See Also

`GetSystemDirectory`, `GetWindowsDir`

### GetWindowsDir (3.1)

```
#include ver.h
```

```
UINT GetWindowsDir(lpszAppDir, lpszPath, cbPath)
LPCSTR lpszAppDir; /* address of Windows directory */
LPSTR lpszPath; /* address of
buffer for path
/
int cbPath;
* size of buffer for path
/
```

The GetWindowsDir function retrieves the path of the Windows directory. This directory contains such files as Windows applications, initialization files, and help files.

GetWindowsDir is used by MS-DOS applications that set up Windows applications; it exists only in the static-link version of the File Installation library. Windows applications should use the GetWindowsDirectory function to determine the Windows directory.

| Parameter  | Description                                                                                                                                                                                                                                       |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpszAppDir | Specifies the current directory in a search for Windows files. If the Windows directory is not on the path, the application must prompt the user for its location and pass that string to the GetWindowsDir function in the lpszAppDir parameter. |
| lpszPath   | Points to the buffer that will receive the null-terminated string containing the path.                                                                                                                                                            |
| cbPath     | Specifies the size, in bytes, of the buffer pointed to by the lpszPath parameter.                                                                                                                                                                 |

#### Returns

The return value is the length of the string copied to the lpszPath parameter, including the terminating null character, if the function is successful. If the return value is greater than the cbPath parameter, it is the size of the buffer required to hold the path. The return value is zero if the function fails.

#### Comments

The path that this function retrieves does not end with a backslash unless the Windows directory is the root directory. For example, if the Windows directory is named WINDOWS on drive C, the path retrieved by this function is C:\WINDOWS. If Windows is installed in the root directory of drive C, the path retrieved is C:\.

After the GetWindowsDir function locates the Windows directory, it caches the location for use by subsequent calls to the function.

#### See Also

GetSystemDir, GetWindowsDirectory

### VerFindFile (3.1)

```
#include ver.h

UINT VerFindFile(flags, lpszFilename, lpszWinDir, lpszAppDir, lpszCurDir, lpuCurDirLen,
                 lpszDestDir, lpuDestDirLen)
    UINT flags; /* source-file flags */
    LPCSTR lpszFilename;
    * address of buffer for file
    /
    LPCSTR lpszWinDir;
    * address of Windows directory
    /
    LPCSTR lpszAppDir;
    * address of application directory
    /
    LPSTR lpszCurDir;
    * address of buffer for current directory
    /
    UINT FAR* lpuCurDirLen;
    * address of buffer size for directory
    /
    LPSTR lpszDestDir;
    * address of buffer for dest. directory
    /
    UINT FAR* lpuDestDirLen;
    * address of size for dest. directory
    /
```

The VerFindFile function determines where to install a file based on whether it locates another version of the file in the system. The values VerFindFile returns are used in a subsequent call to the VerInstallFile function.

| Parameter     | Description                                                                                                                                                                                                                                                                                                                                                         |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| flags         | Contains a bitmask of flags. This parameter can be VFFF_ISSHAREDFILE, which indicates that the source file may be shared by multiple applications. VerFindFile uses this information to determine where the file should be copied. All other values are reserved for future use.                                                                                    |
| lpszFilename  | Points to a null-terminated string specifying the name of the file to be installed. This name should include only the filename and extension, not a path.                                                                                                                                                                                                           |
| lpszWinDir    | Points to a null-terminated string specifying the Windows directory. This string is returned by the GetWindowsDir function. The dynamic-link library (DLL) version of VerFindFile ignores this parameter.                                                                                                                                                           |
| lpszAppDir    | Points to a null-terminated string specifying the drive letter and directory where the installation program is installing a set of related files. If the installation program is installing an application, this is the directory where the application will reside. This directory will also be the application's working directory unless you specify otherwise.  |
| lpszCurDir    | Points to a buffer that receives the path to a current version of the file being installed. The path is a null-terminated string. If a current version is not installed, the buffer will contain the source directory of the file being installed. The buffer must be at least _MAX_PATH bytes long.                                                                |
| lpuCurDirLen  | Points to a null-terminated string specifying the length, in bytes, of the buffer pointed to by lpszCurDir. On return, lpuCurDirLen contains the size, in bytes, of the data returned in lpszCurDir, including the terminating null character. If the buffer is too small to contain all the data, lpuCurDirLen will be greater than the actual size of the buffer. |
| lpszDestDir   | Points to a buffer that receives the path to the installation directory recommended by VerFindFile. The path is a null-terminated string. The buffer must be at least _MAX_PATH bytes long.                                                                                                                                                                         |
| lpuDestDirLen | Points to the length, in bytes, of the buffer pointed to by lpszDestDir. On return, lpuDestDirLen contains the size, in bytes, of the data returned in lpszDestDir,                                                                                                                                                                                                 |

including the terminating null character. If the buffer is too small to contain all the data, `lpuDestDirLen` will be greater than the actual size of the buffer.

#### Returns

The return value is a bitmask that indicates the status of the file, if the function is successful. This value may be one or more of the following:

| Error                         | Meaning                                                                                                                                                                                                                                    |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>VFF_CURNEDEST</code>    | Indicates that the currently installed version of the file is not in the recommended destination.                                                                                                                                          |
| <code>VFF_FILEINUSE</code>    | Indicates that Windows is using the currently installed version of the file; therefore, the file cannot be overwritten or deleted.                                                                                                         |
| <code>VFF_BUFFTOOSMALL</code> | Indicates that at least one of the buffers was too small to contain the corresponding string. An application should check the <code>lpuCurDirLen</code> and <code>lpuDestDirLen</code> parameters to determine which buffer was too small. |

All other values are reserved for future use.

#### Comments

The dynamic-link library (DLL) version of `VerFindFile` searches for a copy of the specified file by using the `OpenFile` function. In the LIB version, the function searches for the file in the Windows directory, the system directory, and then the directories specified by the PATH environment variable.

`VerFindFile` determines the system directory from the specified Windows directory, or it searches the path.

If the flags parameter indicates that the file is private to this application (not `VFFF_ISSHAREDFILE`), `VerFindFile` recommends installing the file in the application's directory. Otherwise, if the system is running a shared copy of Windows, the function recommends installing the file in the Windows directory. If the system is running a private copy of Windows, the function recommends installing the file in the system directory.

#### See Also

[VerInstallFile](#)

## VerInstallFile (3.1)

```
#include ver.h
```

```
DWORD VerInstallFile(flags, lpszSrcFilename, lpszDestFilename, lpszSrcDir, lpszDestDir, lpszCurDir,  
                    lpszTmpFile, lpwTmpFileLen)
```

```
UINT flags; /* source-file flags */
```

```
LPCSTR lpszSrcFilename;  
* address of source filename
```

```
LPCSTR lpszDestFilename;  
* address of destination filename
```

```
LPCSTR lpszSrcDir;  
* address of buffer for source dir. name
```

```
LPCSTR lpszDestDir;  
* address of buffer for dest. dir. name
```

```
LPCSTR lpszCurDir;  
* address of buffer for preexisting dir.
```

```
LPSTR lpszTmpFile;  
* address of buffer for temp. filename
```

```
UINT FAR* lpwTmpFileLen;  
* address of buffer for temp. file size
```

The VerInstallFile function attempts to install a file based on information returned from the VerFindFile function. VerInstallFile decompresses the file with the LZCopy function and checks for errors, such as outdated files.

| Parameter          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                    |                                                                                                                                                                                                                                                                                      |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| flags              | Contains a bitmask of flags. This parameter can be a combination of the following values:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |       |         |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                    |                                                                                                                                                                                                                                                                                      |
|                    | <table><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>VIFF_FORCEINSTALL</td><td>Installs the file regardless of mismatched version numbers. The function will check only for physical errors during installation. If flags includes VIFF_FORCEINSTALL and lpszTmpFileLen is not a pointer to zero, VerInstallFile will skip all version checks of the temporary file and the destination file and rename the temporary file to the name specified by lpszSrcFilename, as long as the temporary file exists in the destination directory, the destination file is not in use, and the user has privileges to delete the destination file and rename the temporary file. The return value from VerInstallFile should be checked for any errors.</td></tr><tr><td>VIFF_DONTDELETEOLD</td><td>Installs the file without deleting the previously installed file, if the previously installed file is not in the destination directory. If the previously installed file is in the destination directory, VerInstallFile replaces it with the new file upon successful installation.</td></tr></tbody></table> | Value | Meaning | VIFF_FORCEINSTALL | Installs the file regardless of mismatched version numbers. The function will check only for physical errors during installation. If flags includes VIFF_FORCEINSTALL and lpszTmpFileLen is not a pointer to zero, VerInstallFile will skip all version checks of the temporary file and the destination file and rename the temporary file to the name specified by lpszSrcFilename, as long as the temporary file exists in the destination directory, the destination file is not in use, and the user has privileges to delete the destination file and rename the temporary file. The return value from VerInstallFile should be checked for any errors. | VIFF_DONTDELETEOLD | Installs the file without deleting the previously installed file, if the previously installed file is not in the destination directory. If the previously installed file is in the destination directory, VerInstallFile replaces it with the new file upon successful installation. |
| Value              | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |       |         |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                    |                                                                                                                                                                                                                                                                                      |
| VIFF_FORCEINSTALL  | Installs the file regardless of mismatched version numbers. The function will check only for physical errors during installation. If flags includes VIFF_FORCEINSTALL and lpszTmpFileLen is not a pointer to zero, VerInstallFile will skip all version checks of the temporary file and the destination file and rename the temporary file to the name specified by lpszSrcFilename, as long as the temporary file exists in the destination directory, the destination file is not in use, and the user has privileges to delete the destination file and rename the temporary file. The return value from VerInstallFile should be checked for any errors.                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |         |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                    |                                                                                                                                                                                                                                                                                      |
| VIFF_DONTDELETEOLD | Installs the file without deleting the previously installed file, if the previously installed file is not in the destination directory. If the previously installed file is in the destination directory, VerInstallFile replaces it with the new file upon successful installation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                    |                                                                                                                                                                                                                                                                                      |
| lpszSrcFilename    | All other values are reserved for future use.<br>Points to the name of the file to be installed. This is the filename in the directory pointed to by lpszSrcDir; the filename should include only the filename and extension, not a path. VerInstallFile opens the source file by using the                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |         |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                    |                                                                                                                                                                                                                                                                                      |



LZOpenFile function. This means it can handle both files as specified and files that have been compressed and renamed by using the /r option with COMPRESS.EXE.

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpzDestFilename | Points to the name VerInstallFile will give the new file upon installation. This filename may be different than the filename in the directory pointed to by lpzSrcFilename. The new name should include only the filename and extension, not a path.                                                                                                                                                                                                                                             |
| lpzSrcDir       | Points to a buffer that contains the directory name where the new file is found.                                                                                                                                                                                                                                                                                                                                                                                                                 |
| lpzDestDir      | Points to a buffer that contains the directory name where the new file should be installed. The <u>VerFindFile</u> function returns this value in the lpzDestDir parameter.                                                                                                                                                                                                                                                                                                                      |
| lpzCurDir       | Points to a buffer that contains the directory name where the preexisting version of this file is found. VerFindFile returns this value in the lpzCurDir parameter. If the filename specified in lpzDestFilename already exists in the lpzCurDir directory and flags does not include VIFF_DONTDELETEOLD, the existing file will be deleted. If lpzCurDir is a pointer to NULL, a previous version of the file does not exist on the system.                                                     |
| lpzTmpFile      | Points to a buffer that should be empty upon the initial call to VerInstallFile. The function fills the buffer with the name of a temporary copy of the source file. The buffer must be at least _MAX_PATH bytes long.                                                                                                                                                                                                                                                                           |
| lpwTmpFileLen   | Points to the length of the buffer pointed to by lpzTmpFile. On return, lpwTmpFileLen contains the size, in bytes, of the data returned in lpzTmpFile, including the terminating null character. If the buffer is too small to contain all the data, lpwTmpFileLen will be greater than the actual size of the buffer.<br>If flags includes VIFF_FORCEINSTALL and lpwTmpFileLen is not a pointer to zero, VerInstallFile will rename the temporary file to the name specified by lpzSrcFilename. |

#### Returns

The return value is a bitmask that indicates exceptions, if the function is successful. This value may be one or more of the following:

| Value          | Meaning                                                                                                                                                                                                                                   |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| VIF_TEMPFILE   | Indicates that the temporary copy of the new file is in the destination directory. The cause of failure is reflected in other flags. Applications should always check whether this bit is set and delete the temporary file, if required. |
| VIF_MISMATCH   | Indicates that the new and preexisting files differ in one or more attributes. This error can be overridden by calling VerInstallFile again with the VIFF_FORCEINSTALL flag.                                                              |
| VIF_SRCOLD     | Indicates that the file to install is older than the preexisting file. This error can be overridden by calling VerInstallFile again with the VIFF_FORCEINSTALL flag.                                                                      |
| VIF_DIFFLANG   | Indicates that the new and preexisting files have different language or code-page values. This error can be overridden by calling VerInstallFile again with the VIFF_FORCEINSTALL flag.                                                   |
| VIF_DIFFCODEPG | Indicates that the new file requires a code page that cannot be displayed by the currently running version of Windows. This error can be overridden by calling VerInstallFile with the VIFF_FORCEINSTALL flag.                            |
| VIF_DIFFTYPE   | Indicates that the new file has a different type, subtype, or operating system than the preexisting file. This error can be overridden by calling VerInstallFile again with the VIFF_FORCEINSTALL flag.                                   |
| VIF_WRITEPROT  | Indicates that the preexisting file is write-protected. The installation program should reset the read-only bit in the destination file before proceeding with the installation.                                                          |
| VIF_FILEINUSE  | Indicates that the preexisting file is in use by Windows and cannot be deleted.                                                                                                                                                           |
| VIF_OUTOFSPACE | Indicates that the function cannot create the temporary file due to insufficient disk space on the destination drive.                                                                                                                     |

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| VIF_ACCESSVIOLATION  | Indicates that a create, delete, or rename operation failed due to an access violation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| VIF_SHARINGVIOLATION | Indicates that a create, delete, or rename operation failed due to a sharing violation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| VIF_CANNOTCREATE     | Indicates that the function cannot create the temporary file. The specific error may be described by another flag.                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| VIF_CANNOTDELETE     | Indicates that the function cannot delete the destination file or cannot delete the existing version of the file located in another directory. If the VIF_TEMPFILE bit is set, the installation failed and the destination file probably cannot be deleted.                                                                                                                                                                                                                                                                                                       |
| VIF_CANNOTRENAME     | Indicates that the function cannot rename the temporary file but already deleted the destination file.                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| VIF_OUTOFMEMORY      | Indicates that the function cannot complete the requested operation due to insufficient memory. Generally, this means the application ran out of memory attempting to expand a compressed file.                                                                                                                                                                                                                                                                                                                                                                   |
| VIF_CANNOTREADSRC    | Indicates that the function cannot read the source file. This could mean that the path was not specified properly, that the file does not exist, or that the file is a compressed file that has been corrupted. To distinguish these conditions, use <a href="#">LZOpenFile</a> to determine whether the file exists. (Do not use the <a href="#">OpenFile</a> function, because it does not correctly translate filenames of compressed files.) Note that VIF_CANNOTREADSRC does not cause either the VIF_ACCESSVIOLATION or VIF_SHARINGVIOLATION bit to be set. |
| VIF_CANNOTREADDST    | Indicates that the function cannot read the destination (existing) files. This prevents the function from examining the file's attributes.                                                                                                                                                                                                                                                                                                                                                                                                                        |
| VIF_BUFFTOOSMALL     | Indicates that the lpwTmpFile buffer was too small to contain the name of the temporary source file. On return, lpwTmpFileLen contains the size of the buffer required to hold the filename.                                                                                                                                                                                                                                                                                                                                                                      |

All other values are reserved for future use.

#### Comments

VerInstallFile is designed for use in an installation program. This function copies a file (specified by lpzSrcFilename) from the installation disk to a temporary file in the destination directory. If necessary, VerInstallFile expands the file by using the functions in LZEXPAND.DLL.

If a preexisting copy of the file exists in the destination directory, VerInstallFile compares the version information of the temporary file to that of the preexisting file. If the preexisting file is more recent than the new version, or if the files' attributes are significantly different, VerInstallFile returns one or more error values. For example, files with different languages would cause VerInstallFile to return VIF\_DIFFLANG.

VerInstallFile leaves the temporary file in the destination directory. If all of the errors are recoverable, the installation program can override them by calling VerInstallFile again with the VIFF\_FORCEINSTALL flag. In this case, lpzSrcFilename should point to the name of the temporary file. Then, VerInstallFile deletes the preexisting file and renames the temporary file to the name specified by lpzSrcFilename. If the VIF\_TEMPFILE bit indicates that a temporary file exists and the application does not force the installation by using the VIFF\_FORCEINSTALL flag, the application must delete the temporary file.

If an installation program attempts to force installation after a nonrecoverable error, such as VIF\_CANNOTREADSRC, VerInstallFile will not install the file.

#### See Also

[VerFindFile](#)

## VerLanguageName (3.1)

```
#include ver.h

UINT VerLanguageName(uLang, lpszLang, cbLang)
UINT uLang;          /* Microsoft language identifier */
LPSTR lpszLang;     /* address of buffer for language string */
/*
UINT cbLang;
* size of buffer
/
```

The VerLanguageName function converts the specified binary Microsoft language identifier into a text representation of the language.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                   |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uLang     | Specifies the binary Microsoft language identifier. For example, VerLanguageName translates 0x040A into Castilian Spanish. If VerLanguageName does not recognize the identifier, the lpszLang parameter will point to a default string, such as "Unknown language". For a complete list of the language identifiers supported by Windows, see the following Comments section. |
| lpszLang  | Points to the buffer to receive the null-terminated string representing the language specified by the uLang parameter.                                                                                                                                                                                                                                                        |
| cbLang    | Indicates the size of the buffer, in bytes, pointed to by lpszLang.                                                                                                                                                                                                                                                                                                           |

### Returns

The return value is the length of the string that represents the language identifier, if the function is successful. This value does not include the null character at the end of the string. If this value is greater than cbLang, the string was truncated to cbLang. The return value is zero if an error occurs. Unknown uLang values do not produce errors.

### Comments

Typically, an installation application uses this function to translate a language identifier returned by the [VerQueryValue](#) function. The text string may be used in a dialog box that asks the user how to proceed in the event of a language conflict.

Windows supports the following language identifiers:

| Value  | Language            |
|--------|---------------------|
| 0x0401 | Arabic              |
| 0x0402 | Bulgarian           |
| 0x0403 | Catalan             |
| 0x0404 | Traditional Chinese |
| 0x0405 | Czech               |
| 0x0406 | Danish              |
| 0x0407 | German              |
| 0x0408 | Greek               |
| 0x0409 | U.S. English        |
| 0x040A | Castilian Spanish   |
| 0x040B | Finnish             |
| 0x040C | French              |
| 0x040D | Hebrew              |
| 0x040E | Hungarian           |
| 0x040F | Icelandic           |
| 0x0410 | Italian             |
| 0x0411 | Japanese            |
| 0x0412 | Korean              |
| 0x0413 | Dutch               |
| 0x0414 | Norwegian - Bokmål  |
| 0x0415 | Polish              |

|        |                           |
|--------|---------------------------|
| 0x0416 | Brazilian Portuguese      |
| 0x0417 | Rhaeto-Romanic            |
| 0x0418 | Romanian                  |
| 0x0419 | Russian                   |
| 0x041A | Croato-Serbian (Latin)    |
| 0x041B | Slovak                    |
| 0x041C | Albanian                  |
| 0x041D | Swedish                   |
| 0x041E | Thai                      |
| 0x041F | Turkish                   |
| 0x0420 | Urdu                      |
| 0x0421 | Bahasa                    |
| 0x0804 | Simplified Chinese        |
| 0x0807 | Swiss German              |
| 0x0809 | U.K. English              |
| 0x080A | Mexican Spanish           |
| 0x080C | Belgian French            |
| 0x0810 | Swiss Italian             |
| 0x0813 | Belgian Dutch             |
| 0x0814 | Norwegian - Nynorsk       |
| 0x0816 | Portuguese                |
| 0x081A | Serbo-Croatian (Cyrillic) |
| 0x0C0C | Canadian French           |
| 0x100C | Swiss French              |

### VerQueryValue (3.1)

#include ver.h

```
BOOL VerQueryValue(lpvBlock, lpszSubBlock, lplpBuffer, lpcb)
const void FAR* lpvBlock;          /* address of buffer for version resource */
LPCSTR lpszSubBlock;              /* address of value to retrieve */
/
VOID FAR* FAR* lplpBuffer;
/* address of buffer for version pointer */
/
UINT FAR* lpcb;
/* address of version-value length buffer */
/
```

The VerQueryValue function returns selected version information from the specified version-information resource. To obtain the appropriate resource, the GetFileVersionInfo function must be called before VerQueryValue.

| Parameter                                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |      |             |   |                                                                                                                                           |                          |                                                                                                                                                                                                                                                                                        |                                          |                                                                                                                                                                                                                                                                                                                                                               |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-------------|---|-------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpvBlock                                 | Points to the buffer containing the version-information resource returned by the <u>GetFileVersionInfo</u> function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |      |             |   |                                                                                                                                           |                          |                                                                                                                                                                                                                                                                                        |                                          |                                                                                                                                                                                                                                                                                                                                                               |
| lpszSubBlock                             | Points to a zero-terminated string specifying which version-information value to retrieve. The string consists of names separated by backslashes (\) and can have one of the following forms:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |      |             |   |                                                                                                                                           |                          |                                                                                                                                                                                                                                                                                        |                                          |                                                                                                                                                                                                                                                                                                                                                               |
|                                          | <table><thead><tr><th>Form</th><th>Description</th></tr></thead><tbody><tr><td>\</td><td>Specifies the root block. The function retrieves a pointer to the <u>VS_FIXEDFILEINFO</u> structure for the version-information resource.</td></tr><tr><td>\VarFileInfo\Translation</td><td>Specifies the translation table in the variable information block. The function retrieves a pointer to an array of language and character-set identifiers. An application uses these identifiers to create the name of an language-specific block in the version-information resource.</td></tr><tr><td>\StringFileInfo\lang-charset\string-name</td><td>Specifies a value in a language-specific block. The lang-charset name is a concatenation of a language and character-set identifier pair found in the translation table for the resource. The lang-charset name must be specified as a hexadecimal string. The string-name name is one of the predefined strings described in the following Comments section.</td></tr></tbody></table> | Form | Description | \ | Specifies the root block. The function retrieves a pointer to the <u>VS_FIXEDFILEINFO</u> structure for the version-information resource. | \VarFileInfo\Translation | Specifies the translation table in the variable information block. The function retrieves a pointer to an array of language and character-set identifiers. An application uses these identifiers to create the name of an language-specific block in the version-information resource. | \StringFileInfo\lang-charset\string-name | Specifies a value in a language-specific block. The lang-charset name is a concatenation of a language and character-set identifier pair found in the translation table for the resource. The lang-charset name must be specified as a hexadecimal string. The string-name name is one of the predefined strings described in the following Comments section. |
| Form                                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |      |             |   |                                                                                                                                           |                          |                                                                                                                                                                                                                                                                                        |                                          |                                                                                                                                                                                                                                                                                                                                                               |
| \                                        | Specifies the root block. The function retrieves a pointer to the <u>VS_FIXEDFILEINFO</u> structure for the version-information resource.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |      |             |   |                                                                                                                                           |                          |                                                                                                                                                                                                                                                                                        |                                          |                                                                                                                                                                                                                                                                                                                                                               |
| \VarFileInfo\Translation                 | Specifies the translation table in the variable information block. The function retrieves a pointer to an array of language and character-set identifiers. An application uses these identifiers to create the name of an language-specific block in the version-information resource.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |      |             |   |                                                                                                                                           |                          |                                                                                                                                                                                                                                                                                        |                                          |                                                                                                                                                                                                                                                                                                                                                               |
| \StringFileInfo\lang-charset\string-name | Specifies a value in a language-specific block. The lang-charset name is a concatenation of a language and character-set identifier pair found in the translation table for the resource. The lang-charset name must be specified as a hexadecimal string. The string-name name is one of the predefined strings described in the following Comments section.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |      |             |   |                                                                                                                                           |                          |                                                                                                                                                                                                                                                                                        |                                          |                                                                                                                                                                                                                                                                                                                                                               |
| lplpBuffer                               | Points to a buffer that receives a pointer to the version-information value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |      |             |   |                                                                                                                                           |                          |                                                                                                                                                                                                                                                                                        |                                          |                                                                                                                                                                                                                                                                                                                                                               |
| lpcb                                     | Points to a buffer that receives the length, in bytes, of the version-information value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |      |             |   |                                                                                                                                           |                          |                                                                                                                                                                                                                                                                                        |                                          |                                                                                                                                                                                                                                                                                                                                                               |

#### Returns

The return value is nonzero if the specified block exists and version information is available. If lpcb is zero, no value is available for the specified version-information name. The return value is zero if the specified name does not exist or the resource pointed to by lpvBlock is not valid.

## Comments

The string-name in the `lpzSubBlock` parameter can be one of the following predefined names:

| Name             | Value                                                                                                                                                                                                                                                                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Comments         | Specifies additional information that should be displayed for diagnostic purposes.                                                                                                                                                                                                                                                                                    |
| CompanyName      | Specifies the company that produced the file--for example, "Microsoft Corporation" or "Standard Microsystems Corporation, Inc.". This string is required.                                                                                                                                                                                                             |
| FileDescription  | Specifies a file description to be presented to users. This string may be displayed in a list box when the user is choosing files to install--for example, "Keyboard Driver for AT-Style Keyboards" or "Microsoft Word for Windows". This string is required.                                                                                                         |
| FileVersion      | Specifies the version number of the file--for example, "3.10" or "5.00.RC2". This string is required.                                                                                                                                                                                                                                                                 |
| InternalName     | Specifies the internal name of the file, if one exists--for example, a module name if the file is a dynamic-link library. If the file has no internal name, this string should be the original filename, without extension. This string is required.                                                                                                                  |
| LegalCopyright   | Specifies all copyright notices that apply to the file. This should include the full text of all notices, legal symbols, copyright dates, and so on--for example, "Copyright Microsoft Corporation 1990-1991". This string is optional.                                                                                                                               |
| LegalTrademarks  | Specifies all trademarks and registered trademarks that apply to the file. This should include the full text of all notices, legal symbols, trademark numbers, and so on--for example, "Windows(TM) is a trademark of Microsoft Corporation". This string is optional.                                                                                                |
| OriginalFilename | Specifies the original name of the file, not including a path. This information enables an application to determine whether a file has been renamed by a user. The format of the name depends on the file system for which the file was created. This string is required.                                                                                             |
| PrivateBuild     | Specifies information about a private version of the file--for example, "Built by TESTER1 on \TESTBED". This string should be present only if the <code>VS_FF_PRIVATEBUILD</code> flag is set in the <code>dwFileFlags</code> member of the <code>VS_FIXEDFILEINFO</code> structure of the root block.                                                                |
| ProductName      | Specifies the name of the product with which the file is distributed--for example, "Microsoft Windows". This string is required.                                                                                                                                                                                                                                      |
| ProductVersion   | Specifies the version of the product with which the file is distributed--for example, "3.10" or "5.00.RC2". This string is required.                                                                                                                                                                                                                                  |
| SpecialBuild     | Specifies how this version of the file differs from the standard version--for example, "Private build for TESTER1 solving mouse problems on M250 and M250E computers". This string should be present only if the <code>VS_FF_SPECIALBUILD</code> flag is set in the <code>dwFileFlags</code> member of the <code>VS_FIXEDFILEINFO</code> structure in the root block. |

## Example

The following example loads the version information for a dynamic-link library and retrieves the company name:

```
BYTE    abData[512];
DWORD   handle;
DWORD   dwSize;
LPBYTE  lpBuffer;
char    szName[512];

dwSize = GetFileVersionInfoSize("c:\\dll\\sample.dll", &handle);
GetFileVersionInfo("c:\\dll\\sample.dll", handle, dwSize, abData);
VerQueryValue(abData, "\\VarFileInfo\\Translation", &lpBuffer, &dwSize);
);

if (dwSize!=0) {
```

```
    wsprintf(szName, "\\StringFileInfo\\%8lx\\CompanyName", &lpBuffer)
;
    VerQueryValue(abData, szName, &lpBuffer, &dwSize);
}
```

See Also

[GetFileVersionInfo](#), [VS\\_FIXEDFILEINFO](#)

### Version Functions (3.1)

|                               |                                                     |
|-------------------------------|-----------------------------------------------------|
| <u>GetFileResource</u>        | Copies a resource into a buffer                     |
| <u>GetFileResourceSize</u>    | Returns the size of a resource                      |
| <u>GetFileVersionInfo</u>     | Returns version information about a specified file  |
| <u>GetFileVersionInfoSize</u> | Returns the size of version information for a file  |
| <u>GetSystemDir</u>           | Returns the path of the Windows system subdirectory |
| <u>GetWindowsDir</u>          | Returns the path of the Windows directory           |
| <u>VerFindFile</u>            | Determines where to install a file                  |
| <u>VerInstallFile</u>         | Installs a file and checks for errors               |
| <u>VerLanguageName</u>        | Converts a binary language identifier into a string |
| <u>VerQueryValue</u>          | Returns version information about a block           |



