ClassAction(Lite) Contents

Welcome to ClassAction Lite by the Crescent Division of Progress Software (Crescent). ClassAction unleashes the power of the Windows API by providing Visual Basic 4.0 developers an object interface to the API. Instead of declarations and function calls, ClassAction is objects with properties and methods.

ClassAction Lite is a limited edition of Crescent's ClassAction. The Lite version contains a subset of the objects found in the standard edition. Also, the Lite version is a Win32 product only whereas ClassAction supports both Win32 and Win16. This helpfile documents the full version of the product. The Objects section shows which objects are found in the Lite version. At the end of this document is a registration form to upgrade to the full copy of ClassAction for $69.00 (US). Mail or Fax your registration today!

For Help on Help, Press **F1**

Hardware and Software Requirements

ClassAction(Lite) has the following hardware and software requirements:

- Any IBM-compatible machine with and 80386 processor or higher
- A 3 ½" floppy drive or CD-ROM Drive
- Microsoft Windows® 95, Windows NT™ Workstation version 3.51 or later
- 8 MB of memory (16 MB or more recommended) if using Windows 95; 16 MB if using Windows NT Workstation
- Visual Basic 4.0 Standard, Professional or Enterprise Edition
- 5 MB of hard-disk space

ClassAction Technical Support

The Crescent technical support staff is ready to help you with problems that you encounter when installing or using ClassAction.

If you need technical support, contact Crescent using any of the following methods:

- By Telephone - Contact Crescents North American technical support staff at: (617) 280-3000 -- Monday through Friday from 9:00 a.m. to 5:00 p.m. EST.

- By FAX - Contact Crescent by FAX at: (617) 280-4025.

- Via BBS - Contact Crescent through our 24-hour bulletin board service at: (617) 280-4221.

- Via CompuServe - Contact Crescent through CompuServe address: 70662,2605 Crescent also maintains a section in the MS Windows Components A+ Forum on CompuServe. To reach the Crescent section, type the following at the CompuServe prompt: GO CRESCENT

- By Electronic Mail - Contact Crescent using the Internet: crescent-support@progress.com

- Via the WWW - View the Crescent Web page at: http://www.progress.com/crescent

- By Mail - Address your correspondence to: Technical Support, Crescent Division, Progress Software Corporation 14 Oak Park Bedford, Massachusetts 01730

Please have your product name, version number, serial number, and system configuration information available so that the Crescent technical support staff can process your support requests as efficiently as possible.

ClassAction Lite Registration

First Name: Last Name:

Title:

Company:

Street or PO Box:

Dept./Suite:

City:

State/Province: Zip/Postal Code:

Country: Telephone:

Fax: E-Mail:

Mail or Fax your registration to:

Crescent Division of Progress Software
ClassAction Lite Registration
14 Oak Park
Bedford, MA 01730

Fax: 617-280-4025

Objects

foBitmap
foBrush
foButton
foColors (Lite)
foComboBox
foDeviceContext
foDrive
foFont
foGlobalMemory
foIcon
foListBox
foLocalMemory
foMetrics (Lite)
foPalette
foPen
foPoint (Lite)
foRectangle (Lite)
foRegion
foSize (Lite)
foSystem (Lite)
foTextBox
foWindow (Lite)

foMetrics Properties

BorderHeight
BorderWidth
CursorHeight
CursorWidth
DblBytesCharsEnabled
DblClickRectHeight
DblClickRectWidth
DebuggingVersion
DialogFrameHeight
DialogFrameWidth
FrameHeight
FrameWidth
FullScreenClientHeight
FullScreenClientWidth
HScrollBarBitmapHeight
HThumbWidth
IconHeight
IconSpacingRectHeight
IconSpacingRectWidth
IconWidth
MenuBarHeight
MenuDropAlignmentLeft
MenuDropAlignmentRight
MinHeight
MinTrackHeight
MinTrackWidth
MinWidth
MouseButtonsSwapped
MousePresent
ScreenHeight
ScreenWidth
TitleBarBitmapHeight
TitleBarBitmapWidth
TitleBarHeight
VScrollBarBitmapHeight
VScrollBarBitmapWidth
VThumbHeight

Debugging Version

Property
DebuggingVersion Long (Read only)

Remarks
TRUE or non-zero if the debugging version of USER.EXE is installed; FALSE, or zero, otherwise.

Example:
Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**DebuggingVersion**

Applies To:
foMetrics

API Reference
Function GetSystemMetrics (flag SM_DEBUG)

foButton

Properties

Remarks
A foButton object represents a button control on a window or dialog box.

Microsoft® Windows® provides dialog boxes and controls to support communications between an application and the user. A button is a control the user can turn on or off to provide input to an application. There are three different types of buttons: check box, radio (Option) button, and push (Command) button.

A check box consists of a square box and text (label), an icon, or a bitmap that indicates a choice the user can make by selecting the button. Applications typically display check boxes in a group to permit the user to choose from a set of related, but independent options.

A radio button consists of a round button and text (label), an icon, or a bitmap that indicates a choice the user can make by selecting the button. An application typically uses radio buttons in a group box to permit the user to choose from a set of related, but mutually exclusive options.

A push button is a rectangle containing text (label), an icon, or a bitmap that indicates what the button does when the user selects it. A push button can be one of two styles: standard or default, as indicated by the Default property. A standard push button is typically used to start an operation. It receives the keyboard focus when the user selects it. A default push button is typically used to indicate the most common or default choice. It is selected by simply pressing ENTER when a dialog box has the input focus.

A foButton object provides access to properties and methods specific to button window controls, and is exposed through the Button property of a foWindow object. For more information about windows, see the foWindow object.

foButton Properties

Default
Value

Dialog Frame Size

Property

DialogFrameHeight	Long (Read only)
DialogFrameWidth	Long (Read only)

Remarks

These properties provide the width and height of window containing a dialog.

Example:

Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**DialogFrameHeight**

Applies To:

foMetrics

API Reference

Function GetSystemMetrics (flags SM_CYDLGFRAME,SM_CXDLGFRAME)

Window Frame Size

Property

FrameHeight	Long (Read only)
FrameWidth	Long (Read only)

Remarks

Width and height of window frame for a window that can be resized

Example:

```
Dim SysObject as New foSystem
Debug.print SysObject.Metrics.FrameHeight
```

Applies To:

foMetrics

API Reference

Function GetSystemMetrics (flag SM_CXFRAME,SM_CYFRAME)

Full Screen Client Area

Property

FullScreenClientHeight	Long (Read only)
FullScreenClientWidth	Long (Read only)

Remarks
Width and height of the client area for a full-screen window.

Example:
Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**FullScreenClientHeight**

Applies To:
foMetrics

API Reference
Function GetSystemMetrics (flag SM_CXFULLSCREEN, SM_CYFULLSCREEN)

IconSpacing

Property

IconSpacingRectHeight	Long (Read only)
IconSpacingRectWidth	Long (Read only)

Remarks

Windows NT only: Width and height of rectangular cell that Program Manager uses to position tiled icons.

Windows 95 only: Dimensions of a grid cell for items in large icon view, in pixels. Each item fits into a rectangle of this size when arranged.

These values are always greater than or equal to the Icon size dimensions. See IconSize

Example:

```
Dim SysObject as New foSystem
Debug.print SysObject.Metrics.DblBytesCharsEnabled
```

Applies To:

foMetrics

API Reference

Function GetSystemMetrics (flag SM_CXICONSPACING, SM_CYICONSPACING)

Icon Size

Property

IconWidth	Long (Read only)
IconHeight	Long (Read only)

Remarks
This property returns the width and height of an icon

Example:
Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**IconHeight**

Applies To:
foMetrics

API Reference
Function GetSystemMetrics (flag SM_CXICON, SM_CYICON)

Menu Bar Height

Property
MenuBarHeight Long (Read only)

Remarks
This property returns the height of single-line menu bar.

Example:
Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**MenuBarHeight**

Applies To:
foMetrics

API Reference
Function GetSystemMetrics (flag SM_CYMENU)

Popup menu alignment

Property

MenuDropAlignmentLeft	Boolean (Read only)
MenuDropAlignmentRight	Boolean (Read only)

Remarks

MenuDropAlignmentRight is set to TRUE, or non-zero if pop-up menus are right-aligned relative to the corresponding menu-bar item. Alternately, MenuDropAlignmentLeft is set to true if menus are left aligned.

Example:

Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**MenuDropAlignmentLeft**

Applies To:

foMetrics

API Reference

Function GetSystemMetrics (flag SM_MENUDROPALIGNMENT)

Minimum window size

Property

MinHeight	Long (Read only)
MinWidth	Long (Read only)

Remarks

A window's size (width and height) is given in pixels. If an application sets a window's width and height to zero, Windows sets the size to the default minimum window size. The properties MinHeight and MinWidth give the default minimum window size.

Example:

Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**MinHeight**

Applies To:

foMetrics

API Reference

Function GetSystemMetrics (flag SM_CXMIN and SM_CYMIN)

Minimum tracking size

Property

MinTrackHeight	Long (Read only)
MinTrackWidth	Long (Read only)

Remarks

These properties return the minimum tracking width and height of a window. The user cannot drag the window frame to a size smaller than these dimensions.

Example:

Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**MinTrackWidth**

Applies To:

foMetrics

API Reference

Function GetSystemMetrics (flag SM_CXMINTRACK, SM_CYMINTRACK)

Mouse Buttons Swapped

Property
MouseButtonsSwapped Long (Read only)

Remarks
TRUE or non-zero if the meanings of the left and right mouse buttons are swapped; FALSE, or zero, otherwise.

Example:
Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**MouseButtonsSwapped**

Applies To:
foMetrics

API Reference
Function GetSystemMetrics (flag SM_SWAPBUTTON)

MousePresent

Property
MousePresent Boolean (Read only)

Remarks
TRUE or non-zero if a mouse is installed; FALSE, or zero, otherwise.

Example:
Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**MousePresent**

Applies To:
foMetrics

API Reference
Function GetSystemMetrics (flag SM_MOUSEPRESENT)

Screen Size

Property

ScreenHeight	Long (Read only)
ScreenWidth	Long (Read only)

Remarks

These properties return the width and height of the screen. An application can size a window so that it is extremely large; however, it should not size a window so that it is larger than the screen. Before setting a window's size, the application should check the width and height of the screen by using these properties.

Example:

Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**ScreenWidth**

Applies To:

foMetrics

API Reference

Function GetSystemMetrics (flag SM_CXSCREEN, SM_CYSCREEN)

Titlebar bitmap size

Property

TitleBarBitmapHeight	Long (Read only)
TitleBarBitmapWidth	Long (Read only)

Remarks

Windows NT only: Width and height of bitmaps contained in title bar.

Windows 95 only: Dimensions of caption buttons, in pixels.

Example:

Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**TitleBarBitmapHeight**

Applies To:

foMetrics

API Reference

Function GetSystemMetrics (flag SM_CXSIZE, SM_CYSIZE)

TitleBarHeight

Property
TitleBarHeight Long (Read only)

Remarks
Height of normal window caption area.

Example:
Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**TitleBarHeight**

Applies To:
foMetrics

API Reference
Function GetSystemMetrics (flag SM_CYCAPTION)

Scrollbar Bitmap Size

Property

VScrollBarBitmapHeight	Long (Read only)
VScrollBarBitmapWidth	Long (Read only)
HScrollBarBitmapHeight	Long (Read only)
HScrollBarBitmapWidth	Long (Read only)

Remarks

Windows 95 only:
VScrollBarBitmapWidth, HScrollBarBitmapHeight
Width of vertical scrollbar and height of horizontal scrollbar, in pixels.

HScrollBarBitmapWidth,VScrollBarBitmapHeight
Width of the arrow bitmap on a horizontal scrollbar and height of the arrow bitmap on a vertical scrollbar.

Windows NT only:
VScrollBarBitmapWidth, VScrollBarBitmapHeight: Width and height of arrow bitmap on vertical scrollbar.

Applies To:
foMetrics

API Reference
Function GetSystemMetrics (flag SM_CXVSCROLL,SM_CYHSCROLL)

Scrollbar thumb size

Property

VThumbHeight	Long (Read only)
HThumbWidth	Long (Read only)

Remarks

Height of vertical scrollbar thumb and width of horizontal scrollbar thumb box.

Example:

Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**VThumbHeight**

Applies To:

foMetrics

API Reference

Function GetSystemMetrics (flag SM_CYVTHUMB, SM_CXHTHUMB)

Double Click Rectangle Dimensions

Property

DblClickRectHeight	Long
DblClickRectWidth	Long

Remarks

Windows NT:
Width and height of a rectangle around the location of a 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.

Windows 95:
Dimensions, in pixels, of the rectangle within which two successive mouse clicks must fall to generate a double-click. (The two clicks must also occur within a specified time.)

Example:

```
Dim SysObject as New foSystem
Debug.print SysObject.Metrics.DblClickRectHeight
```

Applies To:

foMetrics

API Reference

Function GetSystemMetrics (flags SM_CXDOUBLECLK, SM_CYDOUBLECLK)

Double Byte Characters Enabled

Property
DblBytesCharsEnabled Long

Remarks
TRUE or non-zero if the double-byte character set (DBCS) version of USER.EXE is installed; FALSE, or zero otherwise

API Reference
Function GetSystemMetrics (flags SM_DBCSENABLED)

Example:
Dim SysObject as New foSystem
Debug.print SysObject.Metrics.**DblBytesCharsEnabled**

Applies To:
foMetrics

Border Height and Width

Property

BorderHeight	Long
BorderWidth	Long

Remarks
Windows NT only:
Width and height of window border.

Windows 95 only:
Dimensions of a single border, in pixels.

Example:
Dim SysObject as New foSystem
SysObject.Metrics.BorderHeight=12

Applies To:
foMetrics

API Reference
Function GetSystemMetrics (flags SM_CXBORDER, SM_CYBORDER)

Cursor Height And Width

Properties

CursorHeight	Long
CursorWidth	Long

Remarks
Windows NT only:.
Windows 95 only:

Example:

```
Dim SysObject as New foSystem
SysObject.Metrics.CursorHeight=12
```

Applies To:
foMetrics

API Reference
Function GetSystemMetrics (flags)

Copyright Notice

foBitmap Object

Methods Properties

Remarks

A foBitmap object is a graphics object used to create, manipulate (scale, scroll, rotate, and paint), and store images as files on a disk.

A bitmap is one of seven objects that can be selected into a device context. The other six objects are pen, brush, font, region, logical palette, and path. For more information about device contexts, see the foDeviceContext object.

A bitmap is represented by pixels and stored as a collection of bits in which each bit corresponds to one pixel. On color systems, more than one bit corresponds to each pixel. A bitmap usually has a .BMP filename extension.

foBitmap Properties

hBitmap
Dimension
Size

hBitmap

This is the handle to the referenced bitmap.

Handle

A unique numeric value defined by the operating environment and used by a program to identify and access an object, such as a window or control. (handles are integers in Win16 and longs in Win32)

Dimension

Remarks
The Dimension property returns or sets a size object that contains the width and height of a foBitmap object in units of 0.1 mm.

These dimensions are not part of the bitmap itself. However, two cooperating applications could use these dimensions to aid in the scaling of bitmaps that are exchanged through the clipboard or by other means.

GetBitmapDimensionEx
SetBitmapDimensionEx

foBitmap Methods

Attach
Delete
Detach
Load

Delete

The DeleteObject function deletes a logical pen, brush, font, bitmap, region, or palette, freeing all system resources associated with the object. After the object is deleted, the specified handle is no longer valid.
hObject
Identifies a logical pen, brush, font, bitmap, region, or palette.
Return Value
If the function succeeds, the return value is TRUE.
If the given handle is not valid or is currently selected into a device context (DC), the return value is FALSE.
Remarks
Do not delete a drawing object (pen or brush) while it is still selected into a DC.
When a pattern brush is deleted, the bitmap associated with the brush is not deleted. The bitmap must be deleted independently.

Detach Bitmap

Detach removes the association between a bitmap and a bitmap object

Load Bitmap

The LoadBitmap function loads the specified bitmap resource from a module's executable file.

```
HBITMAP LoadBitmap(
    HINSTANCE   hInstance,     // handle of application instance
    LPCTSTR   lpBitmapName  // address of bitmap resource name
   );
```

Parameters

hInstance

Identifies the instance of the module whose executable file contains the bitmap to be loaded.

lpBitmapName

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 MAKEINTRESOURCE macro can be used to create this value.

Return Value

If the function succeeds, the return value is the handle of the specified bitmap.

If the function fails, the return value is NULL.

Remarks

If the bitmap pointed to by the lpBitmapName parameter does not exist or there is insufficient memory to load the bitmap, the function fails.

An application can use the LoadBitmap function to access the predefined bitmaps used by the Win32 API. To do so, the application must set the hInstance parameter to NULL and the lpBitmapName parameter to one of the following values:

OBM_BTNCORNERS OBM_OLD_RESTORE
OBM_BTSIZE 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.

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 application must call the DeleteObject function to delete each bitmap handle returned by the LoadBitmap function.

foBrush Object

Methods Properties

Remarks
A foBrush object is a graphics object used to paint the interior of polygons, ellipses, and paths.

Drawing applications use brushes to paint shapes; word processing applications use brushes to paint rulers; computer-aided design (CAD) applications use brushes to paint the interiors of cross-section views; and spreadsheet applications use brushes to paint the sections of pie chars and the bars in bar graphs.

There are two types of brushes: logical and physical. A logical brush is a description of the ideal bitmap that an application would use to paint shapes. A physical brush is the actual bitmap that a device driver creates based on an application's logical-brush definition. For more information about bitmaps, see the foBitmap object.

A brush is one of seven objects that can be selected into a device context. The other six objects are bitmap, pen, font, region, logical palette, and path. For more information about device contexts, see the foDeviceContext object.

foBrush Methods

Attach
AttachStockBlack
AttachStockDarkGray
AttachStockGray
AttachStockHollow
AttachStockLightGray
AttachStockNull
AttachStockWhite
CreatePattern
CreateSolid
Delete
Detach

Syntax
Object.**Attach**

Remarks
The Attach method always connects a Class Action object with the actual item it represents through a handle.

Example
To cause a new foWindow object to reference a specific window:

```
Dim myWindow as New foWindow
myWindow.Attach Me.hWnd
```

AttachStock

AttachStockBlack
AttachStockDarkGray
AttachStockGray
AttachStockHollow
AttachStockLightGray
AttachStockNull
AttachStockWhite

xxxremove below
Syntax

Parameters

Remarks

Example

Applies To

API Reference

The GetStockObject function retrieves a handle to one of the predefined stock pens, brushes, fonts, or palettes.

```
HGDIOBJ GetStockObject(
    int   fnObject          // type of stock object
  );
```

Parameters
fnObject
Specifies the type of stock object. This parameter can be any one of the following values:

Value	Meaning
BLACK_BRUSH	Black brush.
DKGRAY_BRUSH	Dark gray brush.
GRAY_BRUSH	Gray brush.
HOLLOW_BRUSH	Hollow brush (equivalent to NULL_BRUSH).
LTGRAY_BRUSH	Light gray brush.
NULL_BRUSH	Null brush (equivalent to HOLLOW_BRUSH).
WHITE_BRUSH	White brush.
BLACK_PEN	Black pen.
NULL_PEN	Null pen.
WHITE_PEN	White pen.
ANSI_FIXED_FONT	Windows fixed-pitch (monospace) system font.
ANSI_VAR_FONT	Windows variable-pitch (proportional space) system font.
DEVICE_DEFAULT_FONT	Windows NT only: Device-dependent font.
DEFAULT_GUI_FONT	Windows 95 only: Default font for user interface objects such as menus and dialog boxes.
OEM_FIXED_FONT	Original equipment manufacturer (OEM) dependent fixed-pitch (monospace) font.
SYSTEM_FONT	System font. By default, Windows uses the system font to draw menus, dialog box controls, and text. In Windows versions 3.0 and later, the system font is a proportionally spaced font; earlier versions of Windows used a monospace system font.
SYSTEM_FIXED_FONT	Fixed-pitch (monospace) system font used in Windows versions earlier than 3.0. This stock object is provided for compatibility with earlier versions of Windows.
DEFAULT_PALETTE	Default palette. This palette consists of the static colors in the system palette.

Return Value

If the function succeeds, the return value identifies the logical object requested.

If the function fails, the return value is NULL.

Remarks

Use the DKGRAY_BRUSH, GRAY_BRUSH, and LTGRAY_BRUSH stock objects only in windows with the CS_HREDRAW and CS_VREDRAW 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.

The HOLLOW_BRUSH and NULL_BRUSH stock objects are equivalent.

The font used by the DEFAULT_GUI_FONT stock object could change. Use this stock object when you want to use the font that menus, dialog boxes, and other user interface objects use.

It is not necessary (but it is not harmful) to delete stock objects by calling DeleteObject.

CreatePattern

xxxremove

Syntax

Parameters

Remarks

Example

Applies To

API Reference

Is passed an foBitmap obejct
The CreatePatternBrush function creates a logical brush with the specified bitmap pattern. The bitmap cannot be a DIB section bitmap, which is created by the CreateDIBSection function.

```
HBRUSH CreatePatternBrush(
     HBITMAP   hbmp    // handle of bitmap
   );
```

Parameters
hbmp
Identifies the bitmap to be used to create the logical brush.
Return Value
If the function succeeds, the return value identifies a logical brush.
If the function fails, the return value is NULL.
Remarks
A pattern brush is a bitmap that Windows uses to paint the interiors of filled shapes.
After an application creates a brush by calling CreatePatternBrush, it can select that brush into any device context by calling the SelectObject function.
You can delete a pattern brush without affecting the associated bitmap by using the DeleteObject function. Therefore, you can then use this bitmap to create any number of pattern brushes.
A brush created by using a monochrome (1 bit per pixel) bitmap has the text and background colors of the device context to which it is drawn. Pixels represented by a 0 bit are drawn with the current text color; pixels represented by a 1 bit are drawn with the current background color.
The bitmap identified by hbmp cannot be a DIB section, which is a bitmap created by the CreateDIBSection function. If the bitmap is a DIB section, the CreatePatternBrush function fails.
Windows 95: Creating brushes from bitmaps or DIBs larger than 8x8 pixels is not supported. If a larger bitmap is given, only a portion of the bitmap is used.

CreateSolid
Is passed color as long
The CreateSolidBrush function creates a logical brush that has the specified solid color.

```
HBRUSH CreateSolidBrush(
     COLORREF    crColor          // brush color value
   );
```

Parameters
crColor
Specifies the color of the brush.
Return Value
If the function succeeds, the return value identifies a logical brush.
If the function fails, the return value is NULL.

Remarks
A solid brush is a bitmap that Windows uses to paint the interiors of filled shapes.
After an application creates a brush by calling CreateSolidBrush, it can select that brush into any device context by calling the SelectObject function.

foBrush Properties

hBrush	The handle to the referenced brush.

foColors Object

Properties

Remarks

The foColors object encapsulates the functions that allow a user to set and inspect the colors of the basic system objects such as captions, buttons and scrollbars. foColors is only used in conjunction with the system object, foSystem. The foSystem object includes the method Colors which is of type foColors. System colors for monochrome displays are usually interpreted as shades of gray. All colors are specified in RGB format as long integers. See the VB function RGB.

Example

```
Dim myColor as Long
Dim SystemObject as New foSystem
myColor=rgb(100,100,100)
SystemObject.Colors.ActiveBorderColor=myColor
```

API Function

GetSysColor

foColors Properties

The table lists the foColors object properties. For each property the table gives the property name, the operating system underwhich the property is active and a description of the property. All color properties are expressed as long integers and are in RGB format. See the RGB function in VB.

Property	OS	Description
ActiveBorder	All	Active window border.
ActiveCaption	All	Active window caption.
AppWorkspace	All	Background color of multiple document interface (MDI) applications.
Background	All	Desktop color
ButtonFace	NT	Face shading on push buttons
ButtonHilight	W95	Highlight color for buttons (same as COLOR_3DHILIGHT).
ButtonShadow	All	Edge shading on push buttons
ButtonText	All	Text on push buttons.
CaptionText	All	Text in caption, size box, and scroll bar arrow box.
GrayText	All	Grayed (disabled) text. This color is set to 0 if the current display driver does not support a solid gray color.
Hilight	All	Item(s) selected in a control.
HilightText	All	Text of item(s) selected in a control.
InactiveBorder	All	Inactive window border.
InactiveCaption	All	Inactive window caption.
InactiveCaptionText	All	Color of text in an inactive caption.
Menu	All	Menu background.
MenuText	NT	Text in menus.
Scrollbar	All	Scroll bar gray area.
Window	All	Window background.
WindowFrame	All	Window frame.
WindowText	All	Text in windows.

foComboBox

Methods Properties

Remarks
A foComboBox object represents a combo box control on a window or dialog box.

A combo box is a unique control that combines much of the functionality of a list box and an edit (text box) control. There are three types of combo boxes: simple, drop-down, and drop-down lists boxes.

A combo box consists of a list and a selection field. The list presents the options a user can select and the selection field displays the current selection. In the case of a drop-down list box, the selection field is also an edit control and can be used to enter text not in the list.

A foComboBox object provides access to properties and methods specific to combo box window controls, and is exposed through the ComboBox property of a foWindow object. For more information about windows, see the foWindow object.

foComboBox Properties

Dropped
MaxLength
SelLength
SelStart

foComboBox Methods

FindString

foDeviceContext

Methods Properties

Remarks
A foDeviceContext object is used for drawing in the interface of an application or printed output.

One of the chief features of the Microsoft® Windows application programming interface (API) is device independence. Applications can draw and print output on a variety of devices. A device context defines a set of graphic objects and their associated attributes, and the graphic modes that affect output. The graphic objects include a pen for line drawing, a brush for painting and filling, a bitmap for copying or scrolling parts of the screen, a palette for defining the set of available colors, a region for clipping and other operations, and a path for painting and drawing operations.

There are four types of device contexts: display, printer, memory (or compatible), and information. Each type serves a specific purpose, as described in the following table.

Device context	Purpose
Display	Supports drawing operations on a video display.
Printer	Supports drawing operations on a printer or plotter.
Memory	Supports drawing operations on a bitmap.
Information	Supports the retrieval of device data.

An application can obtain a display device context by retrieving the DeviceContext or ClientDeviceContext property of a foWindow object in which output will appear. An application creates a memory device context by calling the CreateCompatible method of the foDeviceContext object that identifies a device context for a particular device. This method creates a bitmap having a color format compatible with the original device.

foDeviceContext Properties

ArcDirection
CurrentBitmap
CurrentBrush
CurrentFont
CurrentPalette
CurrentPen
CurrentRegion
DeviceCapability
DrawingMode
GraphicsMode
hDc
MapMode
MiterLimit
Parent
PolygonFillMode
TextColor
ViewportExtent
ViewportOrigin
WindowExtent
WindowOrigin

foDeviceContext Methods

AbortPath
Attach
AttachWindow
AttachWindowClient
BeginPath
CloseFigure
CreateCompatible
Delete
Detach
DrawAngleArc
DrawArc
DrawArcTo
DrawBitmap
DrawChord
DrawEllipse
DrawFormattedText
DrawGrayText
DrawIcon
DrawLine
DrawLineTo
DrawPie
DrawPolyBezier
DrawPolyBezierTo
DrawPolgon
DrawPolyline
DrawPolylineTo
DrawRectangle
DrawRoundedRectangle
DrawText
EndPath
FillPath
FillRectangle
FillRegion
FlattenPath
FrameRectangle
InvertRectangle
InvertRegion
MoveTo
OffsetViewportOrigin
OffsetWindowOrigin
PaintRegion
PathToRegion
Restore
Save
ScaleViewportExtent
ScaleWindowExtent
SelectBitmap
SelectBrush
SelectFont
SelectPen
StrokeAndFillPath
StrokePath

UpdateColors
WidenPath

foDrive Object

Properties

Remarks
A foDrive object provides access to removable, fixed, CD-ROM, RAM disk, or network drives.

You can use a foDrive object to retrieve information about a specific drive like the number of **Clusters** and **FreeClusters**.

foDrive Properties

BytesPerSector
Clusters
DriveType
FreeClusters
RootPath
SectorsPerCluster

foFont Object

Methods Properties

Remarks

The foFont object contains information needed to format text for display in the interface of an application or for printed output.

A font is a collection of characters and symbols that share a common design. The three major elements of this design are typeface, style, and size. A typeface is a set of characters that share common characteristics, such as stroke.

A font is one of seven objects that can be selected into a device context. The other six objects are bitmap, brush, pen, region, logical palette, and path. For more information about device contexts, see the foDeviceContext object.

foFont Properties

Height
hFont
Italic
Name
StrikeThrough
Underline
Weight
Width

foFont Methods

Attach
AttachANSIFixed
AttachDefault
AttachOEMFixed
AttachSystem
AttachSystemFIxed
Delete
Detach

foGlobalMemory Object

Methods Properties

Remarks
A foGlobalMemory object represents a block of memory from the global heap.

A process can use the GlobalAlloc and LocalAlloc functions to allocate memory. In the linear 32-bit environment of the Win32 API, the local heap and the global heap are not distinguished. As a result, there is no difference between the memory objects allocated by these functions.

foGlobalMemory Properties

Address
Flags
Size
Valid

Address

Flags
Size
Valid

foGlobalMemory Methods

Allocate
Attach
CopyFrom
CopyFromAddress
CopyTo
CopyToAddress
Free
ReAllocate

foIcon Object

Methods Properties

Remarks
A foIcon object is a picture that consists of a bitmapped image combined with a mask to create transparent areas in the picture.

Windows uses icons throughout the user interface to represent objects such as files, printers, applications, and windows. An application can use a foIcon object to retrieve information about a specific icon, or in conjunction with a foDeviceContext object to draw the icon.

folcon Properties

hIcon

folcon Methods

Attach
Destroy
Detach
Load

Load

See Also
foIcon
foBitmap

foListBox Object

Methods Properties

Remarks

A list box is a control window that contains a list of items from which the user can choose. List box items can be represented by text strings, bitmaps, or both. A list box can provide a scroll bar if it is not large enough to display all the items at once.

A foListBox object provides access to properties and methods specific to list box window controls, and is exposed through the ListBox property of a foWindow object. For more information about windows, see the foWindow object.

foListBox Properties

ItemData
ItemHeight
MaxLength

foListBox Methods

Clear
Dir
FindString

foLocalMemory Object

Methods Properties

Remarks
A foLocalMemory object represents a block of memory from the local heap.

foLocalMemory Properties

Address
Flags
Size
Valid

foLocalMemory Methods

Allocate
Attach
CopyFrom
CopyFromAddress
CopyTo
CopyToAddress
Free
ReAllocate

foMetrics Object

Properties

Remarks
A foMetrics object provides access to the dimensions of various system display elements and system configuration settings.

System metrics are the dimensions of various Windows display elements (Display elements are the parts of a window and the Windows display that appear on the screen.) Typical system metrics include the window border width, icon height, and screen width. System metrics also describe other aspects of the system, such as whether the mouse buttons have been swapped. All system metric properties are read-only.

Applications can also retrieve and set the color of display elements such as menus, scroll bars, and buttons by using the properties of a foColors object.

The foMetrics object is exposed through the Metrics property of a foSystem object. See example below.

Example

```
Dim sysobject as new fosystem
Debug.print sysobject.metrics.framewidth
```

foPalette Object

Methods Properties

Remarks
A foPalette object contains the colors that can be displayed or drawn on an output device.

Color palettes are used by devices that are capable of displaying many colors, but that can only display or draw a subset of these at any given time. For such devices, Windows maintains a palette to track and manage the current colors of the device.

A palette is one of seven objects that can be selected into a device context. The other six objects are bitmap, brush, pen, font, region, and path. For more information about device contexts, see the foDeviceContext object.

foPalette Properties

hPalette

foPalette Methods

Attach
AttachDefault
Delete
Detach
Resize

foPen Object

Methods Properties

Remarks
A foPen object is a graphics object used to draw lines and curves.

Drawing applications use pens to draw freehand lines, straight lines, and curves. Computer-aided design (CAD) applications use pens to draw visible lines, hidden lines, section lines and center lines. Word processing and publishing applications use pens to draw borders and rulers.

A pen is one of seven objects that can be selected into a device context. The other six objects are bitmap, brush, font, region, logical palette, and path. For more information about device contexts, see the foDeviceContext object.

foPen Properties

Color
hPen
Style
Width

foPen Methods

Attach
AttachStockBlack
AttachStockNull
AttachStockWhite
Create
Delete
Detach

foPoint Object

Methods

Remarks

A foPoint object is used to specify and manipulate points. It has two properties X and Y defines the x- and y- coordinates of a point. They are Integers in Win16 and Longs in Win32.

foPoint Methods

Offset
Window

Window Method

Syntax
Set *Object=Object2*.**Window**

Parameters

Object	foWindow
Object2	foPoint

Remarks
Sets the referenced window object to the window containing the Point referenced by the foPoint object.

Applies To
foPoint

foRectangle Object

Methods Properties

Remarks
A foRectangle object is used to specify rectangular areas on the screen or in a window.

Rectangles are used for the cursor clipping region, the invalid portion of the client area, an area for displaying formatted test, or the scroll area. Your applications can also use rectangles to fill, frame, or invert a portion of the client area with a given brush, and to retrieve the corrdinates of a window or a window's client area.

foRectangle Methods

ContainsPoint
Copy
Inflate
Intersect
IsEmpty
IsEqual
Normalize
Offset
SetEmpty
SetRectangle
Subtract
Union

foRectangle Properties

Properties

Top	Long (Win32)	Integer(Win16)
Bottom	Long (Win32)	Integer(Win16)
Left	Long (Win32)	Integer(Win16)
Right	Long (Win32)	Integer(Win16)

TopLeft	foPoint
TopRight	foPoint
BottomLeft	foPoint
BottomRight	foPoint

Remarks

The properties of the foRectangle object are the dimensions of the rectangle that it represents. As shown, the properties are in the form of either the dimension itself or a foPoint object. They may be used interchangeably.

Example:

```
Dim myRect as New foRectangle
myRect.Top=5
myRect.Left=7
Debug.Print myRect.TopLeft.x
```

Applies To:

foRectangle

API Reference

SetRect

Normalize Method

Syntax
Object.**Normalize**

Parameters
Object foRectangle object

Remarks
The Normalize method compares the top and bottom values swapping them if the bottom is greater than the top. The same action is performed on the left and right values. This method is useful when dealing with different mapping modes and inverted rectangles.

Applies To:
foRectangle

SetRectangle Method

Syntax
Object.**SetRectangle** *Left,Top,Right,Bottom*

Parameters

Object	foRectangle object	
Left	Integer (Win16)	Long (Win32)
Top	Integer (Win16)	Long (Win32)
Right	Integer (Win16)	Long (Win32)
Bottom	Integer (Win16)	Long (Win32)

Remarks
The Normalize method compares the top and bottom values swapping them if the bottom is greater than the top. The same action is performed on the left and right values. This method is useful when dealing with different mapping modes and inverted rectangles.

Applies To:
foRectangle

ContainsPoint Method

Syntax
*Result=Object.***ContainsPoint(***myPoint)*

Parameters

Result	Boolean
Object	foRectangle object
myPoint	foPoint object

Remarks
The ContainsPoint method checks to see if the input point is within the boundaries of the rectangle object.

Example:

```
Dim myRect as New foRectangle
myRect.top=5
myRect.Left=......

Dim myPoint as New foPoint
myPoint.x=2

If myRect.ContainsPoint(myPoint) Then
.
.
.
End If
```

Applies To:
foRectangle

API Reference
PtInRect

Copy (Rectangle)

Syntax
Object.**Copy** *Object2*

Parameters

Object,Object2	foRectangle object

Remarks
Copies the passed rectangle into the specified rectangle.

Example:
```
Dim myRect as NewfoRectangle
myRect.Copy OldRect
```

Applies To:
foRectangle

API Reference
SetRect
CopyRect

Inflate Rectangle Method

Syntax
*Object.***Inflate** *dx,dy*

Parameters

Object	foRectangle object	
dx,dy	Long(Win32)	Integer(Win16)

Remarks
The InflateRect function increases or decreases the width and height of the specified rectangle. The InflateRect function adds dx units to the left and right ends of the rectangle and dy units to the top and bottom. The dx and dy parameters are signed values; positive values increase the width and height, and negative values decrease them.

Example:
```
myRect.Inflate 5,4
```

Applies To:
foRectangle

API Reference
InflateRect

Intersect Method

Parameters
Intersect is passed an existing foRectangle object and returns a foRectangle object

Remarks
The Intersect method calculates the intersection of two source rectangles and places the coordinates of the intersection rectangle into the destination rectangle. If the source rectangles do not intersect, an empty rectangle (in which all coordinates are set to zero) is placed into the destination rectangle.

Example:
```
Dim myNewRect as New foRectangle
myNewRect=myOldRectA.Intersect myOldRectB
```

Applies To:
foRectangle

API Reference
IntersectRect

IsEmpty Method

Parameters
IsEmpty accepts an existing foRectnagle object and returns a boolean

Remarks
The IsEmpty method determines whether the specified rectangle is empty. A empty rectangle is one that has no area; that is, the coordinate of the right side is less than or equal to the coordinate of the left side, or the coordinate of the bottom side is less than or equal to the coordinate of the top side.

Example:
```
If myRect.IsEmpty then
:
:
EndIf
```

Applies To:
foRectangle

API Reference
IsRectEmpty

IsEqual Method

Syntax
*Result=Object1.***IsEqual***(Object2)*

Parameters

Result	Boolean
Object1	foRectangle object
Object2	foRectangle object

Remarks
The IsEqual method determines whether the two specified rectangles are equal by comparing the coordinates of their upper-left and lower-right corners.

Example:
```
If MyRect.IsEqual MyOtherRect Then

Endif
```

Applies To:
foRectangle

API Reference
EqualRect

SetEmpty

Parameters
(None)

Remarks
This method sets the coordinates of the referenced rectangle to zero

API Reference
SetRectEmpty

Example:
myRect.SetEmpty

Applies To:
foRectangle

Offset

Parameters
Xoffset, Yoffset Long(Win32) Integer(Win16)

Remarks
The Offset method moves the rectangle by the specified offsets.

API Reference
OffsetRect

Example:
myRect.Offset 12,25

Applies To:
foRectangle

Subtract

Remarks
The Subtract method obtains the coordinates of a rectangle determined by subtracting one rectangle from another. The function only when the rectangles intersect completely in either the x- or y-direction. For example, if myRectA has the coordinates (10,10,100,100) and myRectB has the coordinates (50,50,150,150), the methods sets the coordinates of the resulting rectangle to (10,10,100,100). If myRectA has the coordinates (10,10,100,100) and myRectBhas the coordinates (50,10,150,150), however, the function sets the coordinates of the resulting rectangle to (10,10,50,100).

API Reference
SubtractRect

Example:
Dim myRect as new foRectangle

myRect=myRectA.Subtract myRectB

Applies To:
foRectangle

Union

Parameters
Method is passes an foRectangle object and returns a foRectangle object

Remarks
The UnionRect function creates the union of two rectangles. The union is the smallest rectangle that contains both source rectangles.

API Reference
UnionRect

Example:
Dim myUnionRect as new foRectangle

myUnionRectangle=myRect.Union myOtherRect

Applies To:
foRectangle

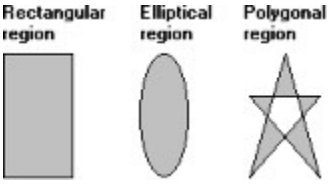
foRegion

Methods Properties

Remarks

A foRegion object is used to specify areas on the screen or in a window.

A foRegion object is a rectangle, polygon, or ellipse (or a combination of two or more of these shapes) that can be filled, painted, inverted, framed, or used to perform hit testing (testing for the cursor location).Following are three types of regions that have been filled and framed.



Every foWindow object has a visible region that defines the window portion visible to the user. The system changes the visible region for the window whenever the window changes size or whenever another window is moves and obscures or exposes part of the window.

A region is one of seven objects that can be selected into a device context. The other six objects are bitmap, brush, pen, font, logical palette, and path. For more information about device contexts, see the foDeviceContext object.

foRegion Properties

Bounds
Complex
hRegion
IsNull
Simple

foRegion Methods

Attach
Combine
ContainsPoint
ContainsRectangle
CreateElliptical
CreatePolygonal
CreateRectangular
CreateRoundRectangular
Delete
Detach
IsEqual
Offset

foSize

Remarks

The foSize object is simply a convenient structure used to specify the size of a rectangular area.. It has two properties X and Y that are both integers in Win16 or long integers in Win32.

foSystem Object

Methods Properties

Remarks
A foSystem object provides access to the configuration information for the current system.

Microsoft® Windows® includes many functions that describe the current system configuration. These functions retrieve a variety of data, such as the computer name, user name, environment variable settings, processor type, system colors, and so on. The foSystem object provides access to many of these settings. Two of its properties, **Colors** and **Metrics**, are actually sub-objects. The **Colors** property is an object of type **foColors**. It provides the ability to inspect and change the color of various display elements. The **Metrics** property is an object of type **foMetrics**. It provides the ability to inspect the dimensions of various display elements such as the menu bar height and title bar height.

At startup, for example, a computer is assigned a name. An application can retrieve this name using the **ComputerName** property. The **ComputerName** property can also be used to change the name. (Note: new names are not assigned until the computer is restarted.)

You can use the **Colors** property to inspect and change the color of window elements such as menus, scroll bars, and buttons.

You can use the **Metrics** property to retrieve the dimensions of various Windows display elements. Typical system metrics inlcude the window border and icon height.

foSystem Properties

ActiveWindow
ComputerName
CurrentDirectory
CursorPosition
DesktopWindow
FocusWindow
ForegroundWindow
LastError
SystemDirectory
TickCount
WindowsDirectory

CurrentDirectory

Syntax
myDirectory=foSystemObject.CurrentDirectory

Parameters
myDirectory String

Remarks
The CurrentDirectory property is used to get or set the current directory for the current process. Each process has a single current directory made up of two parts:
· A disk designator that is either a drive letter followed by a colon, or a server name and share name (\\servername\sharename)
· A directory on the disk designator

API Reference
GetCurrentDirectory, SetCurrentDirectory

Example:

Applies To:
foSystem

TickCount

Syntax
myTime=foSystemObject.TickCount

Parameters
myTime Long Integer

Remarks
The TickCount property retrieves the number of milliseconds that have elapsed since Windows was started. The internal timer wraps around to zero if Windows is run continuously for approximately 49.7 days.

API Reference
GetTickCount

Applies To:
foSystem

ComputerName

Syntax
myName=foSystemObject.ComputerName
foSystemObject.ComputerName=myName

Parameters
myName String

Remarks
The ComputerName method retrieves or sets the computer name of the current system. This name is established at system startup, when it is initialized from the registry. This means that if you change the name, it will not take effect until the system is restarted.

Example:
```
Dim mySys as New foSystem
Dim myName as string
myName="Fred"

mySys.ComputerName=myName
```

Applies To:
foSystem

API Reference
GetComputerName,SetComputername

SystemDirectory (Property)

Syntax
Result=Object.**SystemDirectory**

Parameters

Result	String
Object	foSystem object

Remarks
The SystemDirectory property retrieves the path of the Windows system directory. The system directory contains such files as Windows libraries, drivers, and font files. Applications should not create files in the system directory. If the user is running a shared version of Windows, the application does not have write access to the system directory. Applications should create files only in the directory returned by the WindowsDirectory method.

Applies To:
foSystem

API Reference
GetSystemDirectory

CursorPosition

Syntax
foPointObject=foSystemObject.**CursorPosition**
Set *foSystemObject*.**CursorPosition**=*foPointObject*

Parameters
Sets or assigns an foPoint object

Remarks
The CursorPosition method set or returns the current cursor position.

API Reference
(None)

Example:
Dim myPoint as NewfoPoint
Dim mySys as New foSystem

myPoint.x=3
myPoint.y=9

Set mySys.CursorPosition=myPoint

Applies To:
foSystem

Last Error

Syntax
(Long Integer)=foSystemObject.**LastError**
foSystemObject.**LastError**=*(Long Integer)*

Parameters
Sets or returns a long integer

Remarks
The LastError methods allows the user to set or get the last error. Windows API functions set this value. The Return Value section of each Win32 function's reference page notes the conditions under which the function sets the last-error code. The extended error codes returned by the GetLastError function are not guaranteed to be the same for all operating systems. Note that the last-error code is maintained on a per-thread basis. Multiple threads do not overwrite each other's last-error code. Error codes are 32-bit values (bit 31 is the most significant bit). Bit 29 is reserved for application-defined error codes; no Win32 error code has this bit set. If you are defining an error code for your application, set this bit to one. That indicates that the error code has been defined by an application, and ensures that your error code does not conflict with any error codes defined by the operating system.

API Reference
GetLastError, SetLastError

Example:

Applies To:
foSystem

FocusWindow

Syntax
foWindowObject=foSystemObject.**FocusWindow**
foSystemObject.**FocusWindow=***foWindowObject*

Parameters
Assigns or sets focus to an foWindow object

Remarks
How does this differ from Foregroundwindow?

API Reference

Example:

Applies To:
foSystem

ForegroundWindow

Syntax
foWindowObject=foSystemObject.**ForegroundWindow**
foSystemObject.**ForegroundWindow=***foWindowObject***

Parameters
Assigns or sets an foWindow object

Remarks
The ForegroundWindow method either returns an foWindow object which is the foreground window (the window with which the user is currently working) or puts the thread that created the specified window into the foreground and activates the window. The system assigns a slightly higher priority to the thread that creates the foreground window than it does to other threads. An application should use this method if it wants to put itself into the foreground. A good example is a debugger when it hits a breakpoint. Unlike SetForegroundWindow, the SetActiveWindow function activates a window, but it only brings the window to the foreground if the window is owned by the thread making the SetActiveWindow call. When activated, keyboard input is directed to the window, and various visual cues are changed for the user. The foreground window is the window at the top of the Z order.

API Reference
GetForegroundWindow

Example:
```
Dim myWindow as New foWindow
Dim mySystem as New foSystem

myWindow=mySystem.ForegroundWind
```

Applies To:
foSystem

See Also
ActiveWindow

DesktopWindow

Syntax
Set *foWindowObject=foSystemObject*.**DesktopWindow**

Remarks
The DesktopWindow property returns the handle of the Windows desktop window. The desktop window covers the entire screen. The desktop window is the area on top of which all icons and other windows are painted.

API Reference
GetDesktopWindow

Example:
```
Dim mySys as New foSystem
Dim myDesktop as New foWindow

Set myDesktop=mySys.DesktopWindow
```

Applies To:
foSystem

ActiveWindow

Syntax
Set *foWindowObject=foSystemObject*.**ActiveWindow**
Set *foSystemObject*.**ActiveWindow**=*foWindObject*

Remarks
The ActiveWindow method either sets an foWindowObject to the active window associated with the thread that calls the function.or makes the specified top-level window associated with the thread calling this function the active window. Applications must use this function carefully, because the function allows an application to arbitrarily take over the active window and keyboard focus. Usually, Windows takes care of all activation.

The ActiveWindow method activates a window, but it only brings the window to the foreground if the window is owned by the thread calling the method. The ForegroundWindow method, on the other hand, activates a window and forces the thread that created the window into the foreground.

API Reference
GetActiveWindow
SetActiveWindow

Applies To:
foSystem

WindowsDirectory

Syntax
MyString=foSystemObject.**WindowsDirectory**

Remarks
The WindowsDirectory property retrieves the path of the Windows directory. The Windows directory contains such files as Windows-based applications, initialization files, and Help files.
The Windows directory is the directory where an application should store initialization and help files. If the user is running a shared version of Windows, the Windows directory is guaranteed to be private for each user.

If an application creates other files that it wants to store on a per-user basis, it should place them in the directory specified by the HOMEPATH environment variable. This directory will be different for each user, if so specified by an administrator, via the User Manager administrative tool. HOMEPATH always specifies either the user's home directory, which is guaranteed to be private for each user, or a default directory (for example, C:\USERS\DEFAULT) where the user will have all access.

Example:
```
Dim mySys as New foSystem
Dim myWindowsDir as String

myWindowsDir=mySys.WindowsDirectory
```

Applies To:
foSystem

API Reference
GetWindowsDirectory

foSystem Methods

Colors
MessageBeep
MessageBox
Metrics
SwapMouseButton

MessageBeep

Parameters
MessageBeep is passed an optional variant parameter which should be the number of the sound type desired as enumerated below.

The sound type is identified by an entry in the [sounds] section of the registry. This parameter can be one of the following values:

Value	Sound
0xFFFFFFFF	Standard beep using the computer speaker
MB_ICONASTERISK	SystemAsterisk
MB_ICONEXCLAMATION	SystemExclamation
MB_ICONHAND	SystemHand
MB_ICONQUESTION	SystemQuestion
MB_OK	SystemDefault

Remarks
The MessageBeep function plays a waveform sound. The waveform sound for each sound type is identified by an entry in the [sounds] section of the registry.

After queuing the sound, the MessageBeep function returns control to the calling function 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 through the computer speaker.

The user can disable the warning beep by using the Control Panel Sound application.

Example:
```
Dim mySys as newfoSystem

mySys.MessageBeep MB_OK
```

Applies To:
foSystem

API Reference
MessageBeep

MessageBox

Syntax
Result=foSystemObject.MessageBox(Prompt,Title,Style)

Parameters

Prompt	String	Message or prompt contained in MessageBox
Title	String (Optional)	Title displayed by MessageBox windowOptional
Style	Integer (Optional)	See style flags below
Result	Long Integer	See result codes below

Remarks
The MessageBox method creates, displays, and operates a message box. The message box contains an application-defined message and title, plus any combination of predefined icons and push buttons. Prompt is a string containing the message to be displayed. Tilte is also a string containing the message box title. The style parameter is any combination of the flags shown below.

When an application calls MessageBox and specifies the MB_ICONHAND and MB_SYSTEMMODAL flags, 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 has a Cancel button, the function returns the IDCANCEL value 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.

The return value is zero if there is not enough memory to create the message box.
If the function succeeds, the return value is one of the following menu-item values returned by the dialog box:

Style flags

Value	**Meaning**
MB_ABORTRETRYIGNORE	The message box contains three push buttons: Abort, Retry, and Ignore.
MB_APPLMODAL	The user must respond to the message box before continuing work in the window identified by the hWnd parameter. However, the user can move to the windows of other applications and work in those windows. Depending on the hieracrchy of windows in the application, the user may be able to move to other windows within the application. all child windows of the parent of the message box are automatically disabled, but popup windows are not. MB_APPLMODAL is the default value if neither MB_SYSTEMMODAL nor MB_TASKMODAL is specified.
MB_DEFAULT_DESKTOP_ONLY	The desktop currently receiving input must be a default desktop; otherwise, the function fails. A default desktop is one an application runs on after the user has logged on.

MB_DEFBUTTON1	The first button is the default button. Note that the first button is always the default unless MB_DEFBUTTON2 or MB_DEFBUTTON3 is specified.
MB_DEFBUTTON2	The second button is a default button.
MB_DEFBUTTON3	The third button is a default button.
MB_DEFBUTTON4	The fourth button is a default button.
MB_ICONASTERISK	Same as MB_ICONINFORMATION.
MB_ICONEXCLAMATION	An exclamation-point icon appears in the message box.
MB_ICONHAND	Same as MB_ICONSTOP.
MB_ICONINFORMATION	An icon consisting of a lowercase letter i in a circle appears in the message box.
MB_ICONQUESTION	A question-mark icon appears in the message box.
MB_ICONSTOP	A stop-sign icon appears in the message box.
MB_OK	The message box contains one push button: OK.
MB_OKCANCEL	The message box contains two push buttons: OK and Cancel.
MB_RETRYCANCEL	The message box contains two push buttons: Retry and Cancel.
MB_SERVICE_NOTIFICATION	The caller is a service notifying the user of an event. The function brings up a message box on the current active desktop, even if there is no user logged on to the computer.
MB_SETFOREGROUND	The message box becomes the foreground window. Internally, Windows calls the SetForegroundWindow function for the message box.
MB_SYSTEMMODAL	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 owner window and other windows continue to receive messages resulting from its activation. Use system-modal message boxes 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 hWnd parameter is NULL. Use this flag 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.

Result Codes

Value	**Meaning**
IDABORT	Abort button was selected.
IDCANCEL	Cancel button was selected.
IDIGNORE	Ignore button was selected.
IDNO	No button was selected.
IDOK	OK button was selected.
IDRETRY	Retry button was selected.
IDYES	Yes button was selected.

Applies To:
foSystem

SwapMouseButton

Remarks
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 its buttons affects all applications. This method specifies whether the mouse button meanings are reversed or restored.

Syntax
Swap1=object.SwapMouseButton(Swap2)

Parameters
This method accepts and returns a boolean.

If SWAP2 is TRUE, the left button generates right-button messages and the right button generates left-button messages. If SWAP2 is FALSE, the buttons are restored to their original meanings.

If the meaning of the mouse buttons was reversed previously, before the method was called, the return value, SWAP1, is TRUE.

If the meaning of the mouse buttons was not reversed, the return value, SWAP1, is FALSE.

Applies To:
foSystem

API Reference
SwapMouseButton

foTextBox

Methods Properties

Remarks
A foTextBox object represents an text control on a window or dialog box.

A text control is a rectangular control window typically used in a dialog box to permit the user to enter and edit text.

A foTextBox object provides access to properties and methods specific to text box window controls, and is exposed through the TextBox property of a foWindow object. For more information about windows, see the foWindow object.

A combo box is a control that combines much of the functionality of a text control and a list box. In a combo box, the text control displays the current selection and the list box presents a list of items from which the user can choose. For more information about combo boxes, see the foComboBox object.

foTextBox Properties

CanUndo
Modified
ReadOnly

foTextBox Methods

EmptyUndo
Line
LineCount
ReplaceSelection
Undo

foWindow

Methods Properties Examples

Remarks
A foWindow object is a window, dialog box, or control that makes up part of an application's user interface.

A window in a Microsoft® Windows® application is a rectangular area of the screen where the application displays output and receives input from the user. A window shares the screen with other windows, including those from other applications. Only one window at a time can receive input from the user.

Windows have properties that determine their appearance, such as position and size; and aspects of their behavior, such as whether they are enabled or visible.

Windows also have methods that can be used to manipulate them. For example, you can use the **Move** method to change the window's position and size.

foWindow Properties

Button
ChildWindows
ClassName
ClientDeviceContext
ClientRectangle
ComboBox
Enabled
FirstChild
Font
HasCapture
HasFocus
Height
hWnd
IsWindow
LastActivePopup
Left
ListBox
Parent
Rectangle
Redraw
ShowPopups
Style
Text
TextBox
Top
TopChild
TopLeft
TopMos
UpdateRectangle
UpdateRegion
Visible
Width
WindowState

foWindow Methods

Attach
AttachDesktop
AttachForeground
AttachFromPoint
BringToTop
Cascade
Center
ChildFromPoint
ClientToScreen
Destroy
Detach
DragAcceptFiles
Flash
Hide
Invalidate
InvalidateRectangle
InvalidateRegion
IsChild
Move
Refresh
ScreenToClient
Scroll
SendMessage
SetCapture
SetFocus
Show
Update
ValidateRectangle
ValidateRegion
WinHelp

CaptureMouse

Syntax
[Boolean Variable]=foWIndowObject.**CaptureMouse**
foWIndowObject.**CaptureMouse=***[Boolean expression]*

Remarks
When read, the CaptureMouse property determines in the referenced foWindow object has mouse capture. When set, the CaptureMouse property sets the mouse capture to the referenced foWindowObject. Once a window has captured the mouse, all mouse input is directed to that window, regardless of whether the cursor is within the borders of that window. Only one window at a time can capture the mouse.

Only the foreground window can capture the mouse. When a background window attempts to do so, the window receives messages only for mouse events that occur when the cursor hot spot is within the visible portion of the window. Also, even if the foreground window has captured the mouse, the user can still click another window, bringing it to the foreground.

Example:
myWindow.CaptureMouse=True

Applies To:
foWindow

API Reference
GetCapture
SetCapture
ReleaseCapture

ClassName

Syntax
[String]=foWindowObject.**ClassName**

Remarks
The ClassName property retrieves the name of the class to which the specified window belongs. This is often helpful when examing any of the window collections such as foSystem.Windows collectionor the foWindow.ChildWindows collection

Applies To:
foWindow

See Also
Create
foSystem.Windows collection
foWindow.ChildWindows collection

API Reference
GetClassName

ClientRectangle

Syntax
Set *foRectangleObject=foWindowObject*.**ClientRectangle**

Remarks
The ClientRectangle property allows a use to reference the the referenced foWindow object to retrieves the coordinates of a window's client area by setting it to an foRectangle object. 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).

Applies To:
foWindow

API Reference
GetClientRect

Enabled

Syntax
Boolean=foWindowObject.**Enabled**
foWindowObject.**Enabled**=*Boolean*

Remarks
The Enabled property either determines whether the specified window is enabled for mouse and keyboard input or enables or disables mouse and keyboard input to the specified window or control. When input is disabled, the window does not receive input such as mouse clicks and key presses. When input is enabled, the window receives all input. 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 keyboard 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 use this function to enable or disable a control in a dialog box. A disabled control cannot receive the keyboard focus, nor can a user access it.

Example:
myWindow.Enabled=True

Applies To:
foWindow

API Reference
IsWindowEnabled
EnableWindow

FirstChild

Syntax
foWindowObject=foWindowObject.**FirstChild**

Remarks
The FirstChild property examines the Z order of the child windows associated with the specified parent window and retrieves the child window at the top of the Z order.

Applies To:
foWindow

API Reference
GetTopWindow
GetWindow

hWnd

Syntax
Win 32: *[Long Integer]=foWindowObject*.**hWnd**
Win16: *[Integer]=foWindowObject*.**hWnd**

Remarks
This property provides the hWnd to the referenced foWindow object.

Applies To:
foWindow

IsWindow

Syntax
[Boolean]=foWindowObject.**IsWindow**

Remarks
The IsWindow function determines whether the specified window handle identifies an existing window.

Applies To:
foWindow

See Also
IsWindowEnabled, IsWindowVisible

API Reference
IsWindow

Window Dimensions

Syntax
[Single]=*foWindowObject*.**Left|Top|Width|Height**
foWindowObject.**Left|Top|Width|Height=***[Single]***

Remarks
These properties return the dimensions of the referenced window object in pixels. The Redraw property determines if the window will be redrawn when a dimension is changed.

Example:
Dim ThisWindowas New foWindow
ThisWindow.Attach Me
Debug.print ThisWindow.Width

Applies To:
foWindow

See Also
Attach
Redraw

Parent

Syntax
Set *foWindowObject=foWindObject*.**Parent**
Set *foWindObject*.**Parent=***foWindowObject*

Remarks
The Parent property either changes the parent window of the specified child window or set a reference to the specified child window's parent window.

Applies To:
foWindow

API Reference
SetParent
GetParent

Rectangle

Syntax
Set *foRectangleObject=foWindowObject*.**Rectangle**

Remarks
The Rectangle property returns an foRectangle object based on the dimensions of the bounding rectangle of the specified foWindow object. The dimensions are given in screen coordinates that are relative to the upper-left corner of the screen. Compare with ClientRectangle property which gives only the dimensions of the client area and provides only relative dimensions.

Applies To:
foWindow

See Also
ClientRectangle

API Reference
GetWindowRect

Redraw

Syntax
foWindowObject.**Redraw**=*[Boolean]*
[Boolean]=foWindowObject.**Redraw**

Remarks
This is a flag that determines whether or not a window will be redrawn when a property is altered that will affect its appearence.

Applies To:
foWindow

See Also
Left, Move,Top, Center,Width,Height

API Reference
MoveWindow

ShowPopups

Syntax
foWindowObject.**ShowPopups**=*[Boolean]*

Remarks
The ShowPopups property shows or hides all pop-up windows owned by the referenced window object. The value of the property specifies whether pop-up windows are to be shown or hidden. If TRUE, all hidden pop-up windows are shown. If FALSE, all visible pop-up windows are hidden.
This property controls only windows hidden by a previous calls to ShowPopups. For example, if a pop-up window is hidden by using the property, subsequently setting the property to TRUE does not cause the window to be shown.

Applies To:
foWindow

API Reference
ShowOwnedPopups

Style

Syntax
[Long Integer]=foWindowObject.**Style**
foWindowObject.**Style=***[Long Integer]*

Remarks
The Style property sets or retrieves the window style information as a long integer. It is equivalent to calling the Get and Set WindowLong API function with a flag value of GWL_STYLE.

Applies To:
foWindow

API Reference
GetWindowLong
SetWindowLong

Text

Syntax
[String]=foWindowObject.**Text**
foWindowObject.**Text=***[String]*

Remarks
The Text property sets or gets the text of the specified window's title bar (if it has one) into a buffer. If the specified window is a control, the text of the control is copied. This function cannot retrieve the text of an edit control in another application.

Applies To:
foWindow

See Also
Attach

API Reference
GetWindowText
SetWindowText

TopChild

Syntax
Set *foWindowObject=foWindowObject*.**TopChild**

Remarks
The TopChild property examines the Z order of the child windows associated with the specified parent window and sets the provided foWindow object to the child window at the top of the Z order.

Example:
```
Dim myTopChild as new foWindow
Dim myForm as new foWindow

myForm.Attach Me.hWnd
Set myTopChild=myForm.TopChild
```

Applies To:
foWindow

See Also
FirstChild
[xref]
API Reference
GetTopWindow

TopMost

Syntax
foWindowObject.**TopMost**=TRUE

Remarks
The TopMost property, if true, places the referenced window above all non-topmost windows. The window maintains its topmost position even when it is deactivated. If false then places the window above all non-topmost windows (that is, behind all topmost windows). This flag has no effect if the window is already a non-topmost window.

Applies To:
foWindow

API Reference
SetWindowPos

Visible

Syntax
Boolean=foWindowObject.**Visible**
foWindowObject.**Visible**=*Boolean*

Remarks
The Visible property returns the state of the referenced window visible flag. When set to True, the referenced window is activated and displayed in its current size and position. When set to False, the referenced window is hiddened and another window is activated.

Applies To:
foWindow

API Reference
ShowWindow

WindowState

Syntax
foWindowObject.**WindowState**=vbNormal|vbMaximized|vbMinimized
vbNormal|vbMaximized|vbMinimized=*foWindowObject*.**WindowState**

Remarks
The WindowState property sets or returns the state of the referenced window. It employs the standard VB constants vbNormal, vbMaximized and vbMinimized.

Applies To:
foWindow

API Reference
ShowWindow

Attach

Syntax
foWindowObject.Attach hWnd

Parameters
hWnd is the handle to the window being linked.

Remarks
The Attach method is used to link a foWindow object to a particular instance of a window.

Example:
```
Dim myWindow as New foWindow
myWIndow.Attach Me.hWnd
```

Applies To:
foWindow

AttachDesktop

Syntax
foWindowObject.**AttachDesktop**

Remarks
The GetDesktopWindow function returns the handle of the Windows desktop window. The desktop window covers the entire screen. The desktop window is the area on top of which all icons and other windows are painted.

API Reference
GetDesktopWindow

Applies To:
foWindow

AttachForeground

Syntax
foWindowObject.**AttachForeground**

Remarks
The AttachForeground method sets the referenced window to the foreground window (the window with which the user is currently working). The system assigns a slightly higher priority to the thread that creates the foreground window than it does to other threads.

Applies To:
foWindow

API Reference
GetForegroundWindow

AttachFromPoint

Syntax
foWindowObject.**AttachFromPoint** *foPointObject*

Remarks
The AttachFromPoint method attachs the referenced window to the window that contains the specified point. The method does not retrieve the handle of a hidden or disabled window, even if the point is within the window. An application should use the ChildFromPoint method for a nonrestrictive search.

Applies To:
foWindow

See Also
ChildFromPoint

API Reference
WindowFromPoint

BringToTop

Syntax
foWindowObject.**BringToTop**

Remarks
The BringToTop method brings the specified window to the top of the Z order. If the window is a top-level window, it is activated. If the window is a child window, the top-level parent window associated with the child window is activated.

Use BringWindowToTop to uncover any window that is partially or completely obscured by other windows. Calling this function is similar to setting the TopMost property to True function to change a window's position in the Z order. BringWindowToTop does not make a window a top-level window.

Applies To:
foWindow

See Also
TopMost, TopChild

API Reference
BringWindowToTop

Cascade

Syntax

Remarks
This is a Win32 function only. The CascadeWindows function cascades the specified windows or the child windows of the specified parent window.

Applies To:
foWindow

API Reference
CascadeWindowsByNum
CascadeWindows

Center

Syntax
foWindowObject.**Center** *[Parent]*

Parameters
Parent is an optional parameter of type foWindow

Remarks
The Center method centers the referenced window either on the desktop or, if provided, in the specified window.

API Reference
MoveWindow

Applies To:
foWindow

See Also
Redraw

ChildFromPoint

Syntax
Set *foWindowObject1=foWindowObject*.**ChildFromPoint** foPoint

Remarks
The ChildFromPoint method determines which, if any, of the child windows belonging to a parent window contains the specified point.
If the method succeeds, foWindowObject1 is set to the child window that contains the point, even if the child window is hidden or disabled. If the point lies outside the parent window, the return value is NULL. If the point is within the parent window but not within any child window, foWindowObject1 will reference the parent window.
Windows maintains an internal list, containing the child windows associated with a parent window. The order of the windows in the list depends on the Z order of the child windows. If more than one child window contains the given point, this method will returns the first window in the list that contains the point.

Applies To:
foWindow

See Also
AttachFromPoint

API Reference
ChildWindowFromPoint

ClientToScreen

Syntax
foWindowObject.ClientToScreen foPointObject

Remarks
The ClientToScreen method replaces the client coordinates in the foPoint object with the screen coordinates. The screen coordinates are relative to the upper-left corner of the screen.

API Reference
ClientToScreen

Applies To:
foWindow

See Also
ScreenToClient

Destroy

Syntax
foWindowObject.**Destroy**

Remarks
The Destroy method destroys the given window. The deactivates it and removes the keyboard focus from it. The method also destroys the window's menu, flushes the thread message queue, destroys timers, removes clipboard ownership, and breaks the clipboard viewer chain (if the window is at the top of the viewer chain).

If the given window is a parent or owner window, Destroy automatically destroys the associated child or owned windows when it destroys the parent or owner window. The method first destroys child or owned windows, and then it destroys the parent or owner window.

Destroy also destroys modeless dialog boxes.

Applies To:
foWindow

API Reference
DestroyWindow

Detach

Syntax
foWindowObject.**Detach**

Remarks
The Detach method initializes the referenced foWindow object.

Applies To:
foWindow

See Also
Attach

DragAcceptFiles

Syntax
foWindowObject.**DragAcceptFiles**

Remarks
The DragAcceptFiles method registers whether the referenced window accepts dropped files.

Applies To:
foWindow

API Reference
DragAcceptFiles

Flash

Syntax
foWIndowObject.**Flash** *flag*

Remarks
The Flash method is used to flash a window. Typically, a window is flashed to inform the user that the window requires attention but that it does not currently have the keyboard focus. The window can be either open or minimized (iconic). Flashing a window means changing the appearance of its caption bar as if the window were changing from inactive to active status, or vice versa. The Flash method is always invoked with *flag* set to true and is then called repeatedly at the desired flash rate. The last time the flash method is called, *flag* should be set to false which causes the window to be returned to its original state.

Windows 95: If the application is iconic, the *flag* parameter is NOT ignored. If the *flag* parameter is TRUE, the taskbar icon window flashes activ/inactive. If *flag* is FALSE, the taskbar icon window flashes inactive meaning that it does not change colors. It flashes, as if it were being redrawn, but it does not provide the visual invert clue to the user.

Windows NT: If the window is iconic, FlashWindow simply flashes the icon; the *flag* parameter is ignored for iconic windows.

Applies To:
foWindow

API Reference
FlashWindow

Hide

Syntax
foWindowObject.**Hide**

Remarks
The Hide method hides the referenced window and activates another window. It is equivalent to calling the API function ShowWindow with the parameter SW_HIDE.

Example:
Dim my Window as New foWindow
MyWindow.Attach Me.hWnd
MyWindow.Hide

Applies To:
foWindow

API Reference
ShowWindow

IsChild

Syntax
Boolean=foWindowObjectParent.**IsChild** *foWindowObjectChild*

Remarks
The IsChild function tests whether foWindowObjectChild is a child window or descendant window of foWindowObjectParent. A child window is the direct descendant of a specified parent window if that parent window is in the chain of parent windows; the chain of parent windows leads from the original overlapped or pop-up window to the child window.

Applies To:
foWindow

API Reference
IsChild

Move

Syntax
foWindowObject.**Move** Left,Top,Width,Height

Parameters

Left	Single
Top	Optional
Width	Optional
Height	Optional

Remarks
The Move method changes the position and dimensions of the specified window. For a top-level window, the position and dimensions are relative to the upper-left corner of the screen. For a child window, they are relative to the upper-left corner of the parent window's client area. The action of Move depends on the Redraw property setting. If this property is TRUE, the window is repainted. If the parameter is FALSE, no repainting of any kind occurs. This applies to the client area, the nonclient area (including the title bar and scroll bars), and any part of the parent window uncovered as a result of moving a child window. If this parameter is FALSE, the application must explicitly invalidate or redraw any parts of the window and parent window that need redrawing.

Applies To:
foWindow

See Also
Redraw

API Reference
MoveWindow

Refresh

Syntax
foWindowObject.**Refresh**

Remarks
The Refresh method updates the client area of the specified window by sending a WM_PAINT message to the window if the window's update region is not empty. The function sends a WM_PAINT message directly to the window procedure of the specified window, bypassing the application queue. If the update region is empty, no message is sent.

Applies To:
foWindow

API Reference
UpdateWindow

ScreenToClient

Syntax
foWindowObject.**ScreenToClient** *foPoint*

Remarks
The ScreenToClient method converts the screen coordinates of a specified point on the screen to client coordinates. foPoint contains the screen coordinates to be converted. The method uses the referenced window and the screen coordinates given in the foPoint object to compute client coordinates. It then replaces the screen coordinates with the client coordinates. The new coordinates are relative to the upper-left corner of the specified window's client area. The ScreenToClient function assumes the specified point is in screen coordinates.

Applies To:
foWindow

API Reference
ScreenToClient

Scroll

Syntax

Parameters
X Amount
Y Amount
Scroll Rectangle
Clip Rectangle

Remarks
The Scroll method scrolls the content of the specified window's client area. This function exists for backward compatibility. For new applications, use the ScrollWindowEx function.
Specifies the amount, in device units, of horizontal scrolling. If the window being scrolled has the CS_OWNDC or CS_CLASSDC style, then this parameter uses logical units rather than device units. This parameter must be a negative value to scroll the content of the window to the left.
dy
Specifies the amount, in device units, of vertical scrolling. If the window being scrolled has the CS_OWNDC or CS_CLASSDC style, then this parameter uses logical units rather than device units. This parameter must be a negative value to scroll the content of the window up.
lprcScroll
Points to the RECT structure specifying the portion of the client area to be scrolled. If this parameter is NULL, the entire client area is scrolled.
lprcClip
Points to the RECT structure containing the coordinates of the clipping rectangle. Only device bits within the clipping rectangle are affected. Bits scrolled from the outside of the rectangle to the inside are painted; bits scrolled from the inside of the rectangle to the outside are not painted.
Return Value
If the function succeeds, the return value is TRUE.
If the function fails, the return value is FALSE. To get extended error information, call GetLastError.
Remarks
If the caret is in the window being scrolled, ScrollWindow automatically hides the caret to prevent it from being erased and then restores the caret after the scrolling is finished. The caret position is adjusted accordingly.
The area uncovered by ScrollWindow is not repainted, but it is combined into the window's update region. The application eventually receives a WM_PAINT message notifying it that the region must be repainted. To repaint the uncovered area at the same time the scrolling is in action, 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; invalid (unpainted) areas in the window are also offset. ScrollWindow is faster when lprcScroll is NULL.
If lprcScroll 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 UpdateWindow to repaint the window before calling ScrollWindow.
API Reference

Example:

Applies To:
foWindow

SendMessage

Syntax
Result=foWindowObject.**SendMessage** *message,param1,param2*

Parameters

Result	Long	Message dependant result code
Message	Integer	Specifies the message to be sent.
Param1	Integer	Specifies additional message-specific information.
Param2	Long	Specifies additional message-specific information.

Remarks
The SendMessage method sends the specified message to the referenced window. The method calls the window procedure and does not return until the window procedure has processed the message. The PostMessage function, in contrast, posts a message to a thread's message queue and returns immediately.

Applies To:
foWindow

API Reference
SendMessage

xxxref int vs long in param1 w16 v w32

SetFocus

Syntax
Set *foPreviousWindowObject=foWindowObject*.**SetFocus**

Remarks
The SetFocus function sets the keyboard focus to *foWindowObject*. All subsequent keyboard input is directed to this window. The window, if any, that previously had the keyboard focus loses it. If the method succeeds, *foPreviousWindowObject* is set to the window that previously had the keyboard focus.

See Also
xxxref
GetActiveWindow, GetFocus, SetActiveWindow, SetCapture, SetForegroundWindow

Applies To:
foWindow

API Reference
SetFocus

Show

Syntax
foWindowObject.Show Command

Parameters
Command Optional parameter based on table below.

Remarks
The Show method sets the referenced window's show state based on Command parameter. If Command parameter omitted then methods acts as if it was called with the command SW_RESTORE

SW_HIDE	Hides the window and activates another window.
SW_MAXIMIZE	Maximizes the specififed window.
SW_MINIMIZE	Minimizes the specified window and activates the next top-level window in the Z order.
SW_RESTORE	Activates and displays the window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when restoring a minimized window.
SW_SHOW	Activates the window and displays it in its current size and position.
SW_SHOWDEFAULT	Sets the show state based on the SW_ flag specified in the STARTUPINFO structure passed to the CreateProcess function by the program that started the application. An application should call ShowWindow with this flag to set the initial show state of its main window.
SW_SHOWMAXIMIZED	Activates the window and displays it as a maximized window.
SW_SHOWMINIMIZED	Activates the window and displays it as a minimized window.
SW_SHOWMINNOACTIVE	Displays the window as a minimized window. The active window remains active.
SW_SHOWNA	Displays the window in its current state. The active window remains active.
SW_SHOWNOACTIVATE	Displays a window in its most recent size and position. The active window 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. An application should specify this flag when displaying the window for the first time.

Applies To:
foWindow

API Reference
ShowWindow

WinHelp

Syntax
Return=*foWindowObject*.**WinHelp** Helpfile,Command,Optional

Parameters

Return	Boolean variable indicating if request to open help file was sucessful.
Helpfile	String that contains 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.
Command	One of the commands as describes in the table below
Optional	See command table below

Remarks
The WinHelp method starts Windows Help (WINHELP.EXE) and passes additional data indicating the nature of the help requested by the application. The application specifies the name and, where required, the directory path of the help file to display.

Before closing the window that requested Help, the application must call WinHelp with the uCommand parameter set to HELP_QUIT. Until all applications have done this, Windows Help will not terminate. Note that calling WinHelp with the HELP_QUIT command is not necessary if you used the HELP_CONTEXTPOPUP command to start Help.

The following table shows the possible values for the Command parameter and the corresponding formats of the Optional parameter:

HELP_COMMAND	Executes a Help macro or macro string.	Address of a string that specifies the name of the Help macro(s) to execute. If the string specifies multiple macros names, the names must be separated by semicolons. You must use the short form of the macro name for some macros because Help does not support the long name.
HELP_CONTENTS	Displays the topic specified by the Contents option in the [OPTIONS] section of the .HPJ file. This is for backward compatibility. New applications should provide a .CNT file and use the HELP_FINDER command.	Ignored, set to 0.
HELP_CONTEXT	Displays the topic identified by the specified context identifier defined in the [MAP] section of the .HPJ file.	Unsigned long integer containing the context identifier for the topic.
HELP_CONTEXTPOPUP	Displays, in a pop-up window, the topic identified by the	Unsigned long integer containing the context

	specified context identifier defined in the [MAP] section of the .HPJ file.	identifier for a topic.
HELP_FORCEFILE	Ensures that WinHelp is displaying the correct help file. If the incorrect help file is being displayed, WinHelp opens the correct one; otherwise, there is no action.	Ignored, set to 0.
HELP_HELPONHELP	Displays help on how to use Windows Help, if the WINHELP.HLP file is available.	Ignored, set to 0.
HELP_INDEX	Displays the Index in the Help Topics dialog box. This command is for backward compatibility. New applications should use the HELP_FINDER command.	Ignored, set to 0.
HELP_KEY	Displays the topic in the keyword table that matches the specified keyword, if there is an exact match. If there is more than one match, displays the Index with the topics listed in the Topics Found list box.	Address of a keyword string.
HELP_MULTIKEY	Displays the topic specified by a keyword in an alternative keyword table.	Address of a MULTIKEYHELP structure that specifies a table footnote character and a keyword.
HELP_PARTIALKEY	Displays the topic in the keyword table that matches the specified keyword, if there is an exact match. If there is more than one match, displays the Index tab. To display the Index without passing a keyword (the third result), you should use a pointer to an empty string.	Address of a keyword string.
HELP_QUIT	Informs the Help application that it is no longer needed. If no other applications have asked for Help, Windows closes the Help application.	Ignored, set to 0.
HELP_SETCONTENTS	Specifies the Contents topic. The Help application displays this topic when the user clicks the Contents button.	Unsigned long integer containing the context identifier for the Contents topic.
HELP_SETINDEX	Specifies a keyword table to be displayed in the Index of the Help Topics dialog box.	Unsigned long integer containing the context identifier for the Index topic.
HELP_SETWINPOS	Displays the Help window, if it is minimized or in memory, and sets its size and position as specified.	Address of a HELPWININFO structure that specifies the size and position of either a primary or secondary Help window

.

API Reference

Example:

Applies To:
foWindow

Attach

Applies To
foBitmap

FindString

Applies To:
foComboBox

Error Handling

Class Action handles errors in a way that is helpful to the VB developer. When using the Win32 API directly, an engineer must inspect individual function result codes and then, if needed, call the GetLastError to determine if and then what error occurred. Class Action automatically traps all errors and then raises an error event. Youll find that the VB error object has had its properties appropriately set including number, source and description. The number of the error raised will be the API error code offset by the VBObjectError constant. The source property will generally identify the Class Action object in which the error occurred. The description property will generally contain a helpful description.

Obtaining Technical Support

The Crescent technical support staff is ready to help you with problems that you encounter when installing or using Class Action.

How to reach Technical Support

By Telephone	617-280-3000 Monday through Friday 9:00 a.m. to 5:00 p.m. EST
By Fax	617-280-4025
Via BBS	617-280-4221 24 hour service
Compuserve	70662,2605
Internet	crescent-support@progress

Using ClassAction

Using ClassAction is a snap!. The ClassAction installation program installs and registers the necessary dynamic link libraries (DLL).

To access ClassAction in a VB4 project, do the following:

1. Start Visual Basic and load a project.
2. Select **References** command from the **Tools** menu in VB4.
3. Locate and check the Crescent Foundation Objects entry. Then click OK.
4. Save your project.

The ClassAction DLL is registered for use in your project. Now you can dimension variables as any of the objects described in this help file.

For example, to display the computer name you can use the foSystem object:

```
Dim SystemObject As New foSystem
Debug.Print SystemObject.ComputerName
```

To learn more about the objects available in ClassAction see the ClassAction Objects topic. And look in the Samples directory for some helpful sample VB4 projects.

Notation Conventions

The following document conventions are used throughout this helpfile.

Convention	Description
Bold text	Bold letters indicate a specific term or punctuation mark intended to be used literally: language functions or keywords (such as DrawPenDataEx or switch), MS-DOS® commands, and command-line options. You must type these terms and punctuation marks exactly as shown. The use of uppercase or lowercase letters is usually, but not always, significant. For example, you can invoke the C compiler by typing either CL, cl, or Cl at the MS-DOS prompt.
( )	In syntax statements, parentheses enclose one or more parameters that you pass to a function.
Italic text	Italic text indicates a placeholder; you are expected to provide an actual value. For example, in the following syntax the placeholder lpszRecogName represents a pointer to the filename of a recognizer:InstallRecognizer(lpszRecogName);New terms pertaining to pen-based computing also appear in italics where they are first introduced or defined in the text. Such terms are also listed in the glossary.
Monospace text if(!RegisterClass(LPWNDCLASS)&wc)) . . .else	Code examples are displayed in a nonproportional typeface. A vertical ellipsis in a program example indicates that a portion of the program has been omitted.
. . .	A horizontal ellipsis following an item indicates that more items having the same form may appear.
[[ ]]	Double brackets enclose optional fields or parameters in command lines or syntax statements.
|	A vertical bar indicates that you can enter one of the entries shown on either side of the bar. In symbol graphs, a vertical bar indicates the possible character choices.
{ }	Braces indicate that you must specify one of the enclosed items.

Syntax

Parameters

Remarks

Example

Applies To

API Reference