This file contains context-sensitive help topics for Developer Studio.

Processes your input and carries out any default action, such as closing the dialog box.

Cancels your input and carries out the default action, such as closing the dialog box.

Closes the dialog box.

Opens a topic which provides additional information about this dialog box.

Opens a dialog you can use to browse to the location of the file or files you wish to open.

Displays the name of a file that contains one or more macros. Select the file containing the macro you want, or create a new file by clicking the Options button, then the New File button.

Displays the names of all macros in the selected macro file. Select the macro you want to run or edit, or create a new macro by entering a name for the macro in the Macro Name box.

Creates a new macro file. When you click New File, Visual C++ enters the file's name and description in the New Macro File dialog box.

Runs the selected macro. If no macro file is selected, you can either: (1) create a new file by clicking the Options button, then the New File button, or (2) load an existing macro file into Visual C++ from the Add-ins and Macro Files tab in the Customize dialog box (Tools menu, Customize command).

Changes or creates a macro. If you select a macro name from the list of names in the Macro Name box, when you click Edit, Visual C++ opens that macro in the editor. However, if you enter a new macro name in the Macro Name box, when you click Edit, Visual C++ creates that macro and opens it in the editor.

Records a macro. When you click Record, Visual C++ displays the Add Macro dialog box in which you add a description of the macro. After adding the description, click OK to begin recording. Visual C++ appends the new macro to the end of the selected macro file and adds the description at the beginning of the file.

Displays additional macro options, such as assigning the macro to a toolbar button or key sequence, or loading the macro file into Visual C++ Developer Studio.

Assigns a macro to a menu or toolbar button.

Assigns a macro to a key sequence.

Displays the add-ins and macro files that are available and unavailable. Add-ins and macro files that do not have a check mark are not loaded and are not available. Select the check box next to an add-in or macro file to load it and make it available.

Explains what the selected macro does. You provide this description when you record or edit a new macro. Visual C++ puts this description at the beginning of the selected macro file.

Displays help about designing, recording, editing, running, debugging, deleting, or renaming macros. Note! Clicking this button dismisses the Macro dialog box.

Lists macro files and add-ins that are available and unavailable. When selected, macro files and add-ins are loaded and available.

Allows you to browse for macro files or add-ins that you want to include in the Add-ins and macro files list.

Describes the purpose of the selected macro file or add-in.

Displays the name of the macro.

Describes the purpose of the macro.

Select this option if you want Visual C++ to prompt you to reload a macro file after you change a macro in it.

Select this option to always reload a macro file after you have changed a macro in it.

Select this option to never reload a macro file after you have changed a macro in it. However, in the Macro editor, you can always reload a macro file by clicking the right mouse button and then clicking ‘‘Reload Macro File.’’

When selected, prevents this dialog box from reappearing.

Type a name for the folder you want created. Folders created in ClassView store iconic representations of classes, not actual files. To create folders for file storage, use FileView.

For dual interfaces, **HRESULT** is always the return type. For custom interfaces, you can select a return type from the list. **HRESULT** is still recommended, because it provides a standard way to return errors.

Enter a name for the method. Do not include a semi-colon. The complete method declaration is displayed in the Implementation area.

Enter any parameters, separated by a comma, that you want the new method to take.

This read-only field displays the complete method implementation, including the values from Return Type, Method Name, Parameters, and any attributes you define in the Edit Attributes dialog box. To enter attributes, click the Attributes button. You can continue to add or modify attributes until you close the Add Method dialog box.

After typing a name for the new method, click this button to display the Edit Attributes dialog box, where you enter the Name and Value for the new method’s attributes.

Created on the fly in CLS.HID. Not needed after all; aliased to Cls_HIDC_COMBO_RETTYPE.

For custom and dual interfaces, either choose from the drop-down list or enter your own type for this property. Use the property type to pass a value by means of the parameter list, rather than the return type.

Enter the name you want to give this new property.

Specify any parameters you want this property to take. The Implementation area reflects the parameters you enter here.

This read-only field displays the complete property implementation, including the values from Return Type, Property Type, Property Name, Parameters, Function Type, and any attributes you define in the Edit Attributes dialog box. To enter attributes, click the Attributes button. You can continue to add or modify attributes until you close the Add Property dialog box.

After typing a property declaration, click this button to display the Edit Attributes dialog box, where you enter the name and value for the new property's attributes.

Selecting this option makes the property readable, that is, creates the Get method for retrieving this property from the object. You must select either Get, Put, or both.

Selecting this option makes this property writable, that is, it creates the Put method for setting, or “putting,” this property of the object. You must select either Get, Put, or both. If you select this option, you can choose from two ways to implement the method: with a PropPut, or with a PropPutRef.

The PropPut function returns a copy of the object. This is the default and the most common way to make the property writable.

The PropPutRef function returns a reference to the object, rather than returning the object itself. Consider using this option for objects, such as large **structs** or **arrays**, that may have initialization overhead.

For dispinterfaces, select from the predefined list to specify a property type. Use the property type to pass a value by means of the parameter list, rather than the return type.

This read-only field displays the name of the new method whose attributes you specify by choosing from the Attribute grid.

Select an attribute from the Name drop-down list box. Then, either select from a list of possible choices for Value, or enter your own. Only certain attributes take values. The **helpstring** and **id** attributes are added by default. All Attributes in this list are defined in the Microsoft Interface Definition Language (MIDL) reference.

This read-only field displays the name of the new property whose attributes you define by choosing from the Attribute grid.

Select the language whose elements you want WizardBar to track. Only languages implemented in the active project appear in this list.

“Apphelp” topics start here ====

Selected Object Properties: Attributes

Attributes

This tab displays the defined attributes for the selected object. That object may be a method, a property, an implemented method or property, or a function. The title bar of the Properties sheet displays the type of object. With your cursor in the Attributes field, you can type a letter to jump to any attribute in the list that starts with the letter you've typed. However, you cannot enter text into this field.

Class Properties: General

Name
Displays the name of the selected class.

Base Class
Displays the base class from which the selected class is derived.

Folder Properties: General

Folder Name
Displays the name for this user-defined folder. You can use this property page to edit the folder name.

Project Folder Properties: General

Project

No properties are available for the project folder from ClassView. To view project properties, switch to FileView.

Data Properties: General

Name
Displays the member variable name.

Type
Displays the member variable type.

Function Properties: General

Name
Displays the function name.

Return Type
Identifies the type of value that the implemented method returns.

Parameters
Lists the types of parameters the function takes, in declaration order.

Interface Properties: General

Name
Displays the interface name.

Derived From
Displays the interface from which this interface is derived, for example, IDispatch or IUnknown.

Attributes
Displays any attributes defined for this interface. You can jump through this list by typing the first letter of the attribute into the list box.

Property Properties: General

Name
Displays the name of the property.

Return Type
Identifies the type of value that the property returns.

Parameters
Lists the types of parameters the property takes, if any, in declaration order.

Property Properties: General

Name
Displays the name of the implemented property.

Return Type
Identifies the type of value that the property returns.

Parameters
Lists the types of parameters the property takes, if any, in declaration order.

Method Properties: General

Name
Displays the method name.

Return Type
Identifies the type of value that the method returns.

Parameters
Lists the types of parameters the method takes, if any, in declaration order.

Method Properties: General

Name
Displays the name of the implemented method.

Return Type
Identifies the type of value that the method returns.

Parameters
Lists the types of parameters the method takes, if any, in declaration order.

Specifies the library on which the new class will be based, or whether this will be a user-derived (Generic) class. Choose from the drop-down to change the default selection, which is based on the project to which you are adding the class.

Never shows up

Enter the type (for example, **INT**) that the member function returns.

Enter the name of the member function, followed by a parentheses-enclosed list of the names and types of any formal parameters. You can declare a constant by adding a "const" after the parameter list, for example "MyFunction (int i) **const**."

When selected, declares the function as a **public** member function that can be accessed by any other function in the program.

When selected, declares the function as a **protected** member function that can be accessed by member functions of the same class or any class derived from the same class.

When selected, declares the function as a **private** member function that can be accessed by member functions of the same class. This action is not recommended if the class may be used as a base class.

When selected, declares this function as a **static** function.

When selected, declares this function as a **virtual** function.

Enter the type for the variable, for example, **float**.

Enter the name you want to give the variable.

When selected, declares the function as a **public** member variable that can be accessed by any other function in the program.

When selected, declares the function as a **protected** member variable that can be accessed by member functions of the same class or any class derived from the same class.

When selected, declares the function as a **private** member variable that can be accessed by member functions of the same class. This action is not recommended if the class may be used as a base.

Same description as IDD_GEN_CLASS instance.

Uses the same text as that for IDD_GEN_CLASS

Select this button to create a class with a dual interface (its **vtable** has custom interface functions plus late-binding **IDispatch** methods). Allows both COM clients and Automation controllers to access the object.

Select this button to create a class with a custom interface (its **vtable** has custom interface functions). A custom interface can be faster than a dual interface, especially across process boundaries.

Set the number of interfaces for the class.

Displays the interface names. To edit a name, choose the Edit button.

Click this button to edit an interface name.

Select this option to make the class aggregatable.

The generated name of the header file where you want to insert the declaration. Edit the name if you like, or select Browse to specify a different file.

The generated name of the .CPP file where you want to insert the class implementation. Edit the name if you like, or select Browse to specify a different file.

Click this button to use a file dialog box to locate a header file where you want to insert the declaration.

Click this button to use a file dialog box to locate a .CPP file where you want to insert the function.

The file extension associated with documents created by your program. When entering the file extension, do not include the period. Entering a file extension allows the Explorer to interact more directly with your program, for example launching it on double-click and printing your program's documents when the document icons are dropped on a printer icon.

This ID is used to label your document type in the system registry.

The type of document under which this file can be grouped. This option then becomes available from the “Save as type:” drop-down in the Windows File Save As dialog box.

By default this box displays the extension you enter in the File Extension box and names the file type after your program. For example, if your project is named Foo, and the file extension is .foo, the filter name is: “Foo Files (*.foo).” You can edit the text in this box.

The name that appears in the **New** dialog box if there is more than one new document template. If your program is an Automation server this name is used as the short name of your Automation object.

If your program is an Automation server, this name is used as the long name of your Automation object. It is also used as the file type name in the system registry.

Displays the generated object name, which is used as the base name for the CoClass, Type name, type ID, and interface names. You can edit the name.

The name of the coclass that contains a list of interfaces supported by the object.

A description string for the object that goes into the registry.

A name that containers can use instead of the CLSID of the object.

The names of the interfaces supported by the object.

Enter the name for the class or form you want to add. The filename for the associated .cpp and .h files, and the ID for the dialog associated with the form, are initially generated from this name.

The generated name of the implementation file associated with the class or form. You can change this name by clicking the Change button.

Click to change the file name or names.

Lists the base classes from which the class is to be derived. Enter class names in the first column. In the second column, choose the type of inheritance (**public**, **private**, or **protected**) for each base class.

Use items in this group box to define the type of new form you want added to the active project.

Use text from IDD_GEN_CLASS.

Uses the same text as for IDD_GEN_CLASS.

Uses text from IDD_GEN_CLASS.

The MFC base class from which you want to derive the form.

The resource ID for the dialog associated with the form. You can create your own (new) dialog resource, or reuse an existing one. If the base class is derived from a form view, the dialog properties must include: **WS_CHILD** = On; **WS_BORDER**=Off; **WS_VISIBLE**=Off; **WS_CAPTION**=Off. All of these styles can be set from the properties page for the dialog. Note that not all resource IDs that may populate this list are guaranteed to have the correct properties for a form.

Use options in this group box to specify Automation options for the form. Available only for Automation-capable classes, that is, those derived from **CCmdTarget**.

Specifies that no Automation will be provided for the form.

Specifies that the form enables objects which can be accessed by Automation clients, such as Microsoft Visual Basic and Microsoft Excel.

Use this group box to specify or change options for the document associated with your **CFormView**-based form.

Specifies the document template to use with the new form. You can choose from the list of **CDocument**-based classes in your project, or create a new class by clicking New. Not valid for forms based on **CDialog**.

Click this to specify a new **CDocument**-based class to associate with the form.

The auto-generated extension for the form, based on the form name. C++ uses the extension specified here to determine which document template to use on File|Open.

Click to change the auto-generated extension, for example to use your own naming convention. In an application with multiple forms, each form should have a unique extension.

Lists the known MFC virtual functions for the selected class.

Lists the virtual functions that are already overriden for the selected class.

Choose this button to jump to the selected existing code, opening the file where it resides if necessary.

Select this button to add the selected New Virtual Function to the class specified in the title bar of the dialog. A skeleton function is added to the .cpp file, and a reference to the function is added to the .h file.

Select this button to both generate skeleton code for the new virtual function, and open the .cpp file with your cursor at the location of the function.

Choosing this button provides information on the selected Windows message.

Provides a description of the item selected.

Displays available dialog classes, and the IDs of existing dialog controls for which you can add a Windows message handler. The contents of the New Windows messages/ events list box changes to reflect your selection in this list box.

Displays Windows messages that already have handlers created for them. Choose Edit Existing to open a source editor window with the existing handler code.

Select from the drop-down to filter for messages that apply only to a certain class or window type. For example, if you select Child Window, only those messages valid for child window types appear in the New Windows messages/events list.

Type in the name of the message handler you want to create.

Choose a message handler from the list of current possible message handlers for the selected class or object.

This adds the message handler to the selected class or object.

This adds the message handler to the selected class or object, and goes to the class or object for editing.

Same as IDD_NEWVIRTUAL

Same as NEW_VIRTUAL.

Select a file or files from the list to include.

IDD_PARSE_DLGHIDC_PARSE_TYPE Outdated dialog.

IDD_PARSE_DLGHIDC_PARSE_FILE Apparently we don’t use this dialog.

IDD_PARSE_DLGHIDC_PARSE_LIST Outdated dialog.

Shows a list of current breakpoints. Use the checkboxes to enable or disable specific breakpoints.

Use this button to go to a source window containing the code where the breakpoint is set.

Deletes the breakpoint selected in the Breakpoints list.

Deletes all Breakpoints currently set.

Lists the instances in which an identical symbol occurs. Choose one of the instances to examine.

Displays status information on each thread.

A unique identifier for each thread. When you set focus on a thread, it is displayed in the Thread List with an asterisk next to its identifier.

The number of times the thread has been suspended. This value is decremented each time the thread is resumed. The thread runs only if this value equals zero.

An integer value from 0 to 31 that determines the priority in which the threads will run.

Displays the current address of the thread. The Name and Address buttons determine whether this appears as a function name or as an address value.

Displays the Location as a function name, if the name is known to the debugger.

Displays the Location as an address value instead of a function name.

Increments the suspend count of the thread selected in the Thread List. A thread is suspended when its count goes from 0 to 1.

Decrements the suspend count of the suspended thread selected in the Thread List. A thread resumes when its count goes from 1 to 0.

Switches the focus to the thread selected in the Threads List.

Displays the unique number of the exception. System exceptions are defined in WINBASE.H with the prefix of **EXCEPTION** (for example **EXCEPTION_ACCESS_VIOLATION**).

Lists the optional name to be displayed in the Exception list for the exception.

Indicates the action to take when the debugger is notified of an exception. It can take two actions: Stop Always or Stop If Not Handled.

Indicates the action to take when the debugger is notified of an exception. It can take two actions: Stop Always or Stop If Not Handled.

Displays the list of system exceptions that you want to handle. You can modify this list, deleting system exceptions or adding your own. This information is saved in the *project*.dsw file (where *project* represents your project name) and persists with the project. You can select multiple exceptions.

Adds the exception, as specified in the Number and Name text boxes, to the Exceptions list, along with an optional action. The default action, if none is specified, is Stop If Not Handled.

Deletes the selected exception(s) from the Exceptions list. The debugger still handles deleted exceptions with the action Stop If Not Handled.

Accepts changes made to the highlighted exception(s) in the Exceptions list. A change might include a different action, for example.

Restores all default system exceptions to the Exceptions list without disturbing any of the user-defined exceptions that have been added.

Same as IDD_AMBIG_SRCLINES.

Enter the beginning address for the memory block you want to display. The address can be either a numeric address or a symbolic expression.

Specifies the type of computer you want to connect to.

Specifies the type of communication transport you want to use to connect your computers.

Opens the dialog box you use to specify communications settings for the transport you have chosen.

Enabled only if you select Power Macintosh as the platform. Select this button to debug an active process.

Enter the name of the **WndProc** you want to break on, or select a **WndProc** name from the dropdown list.

Enter the message that you want to break on, or select a message from the dropdown list.

Select the scope or context containing the variables you want to view.

Specify a local module name (or use the Browse button).

When checked, the Find Local Module to asks you to locate additional DLLs after the current DLL or EXE.

Find local modules using the Browse for Local Module dialog to find an EXE or DLL.

Indicates the variable or expression to display or modify.

Displays the expression from the Expression text box under Name, and its value (or values for an **array**, **object**, or **structure**) under Value. If you selected hexadecimal display in the Tools Option window Debug tab, you must enter your input in hexadeciaml or use the prefix `0n` (zero-n) to indicate that the number is a decimal (for example: `0n1000`) **.**

Calculates the value of an expression in the Expression text box, and displays the result into the Current Value text box.

Adds the variable or expression to the Watch window. If the Watch window is not displayed, it also displays the window.

A variable or expression that determines whether the debugger stops at the breakpoint. The expression can be a **Boolean** expression, which evaluates to true or false, or a non-**Boolean** expression such as a+b/c. If you specify a **Boolean** expression, the execution stops at the breakpoint if the condition evaluates to true. If you specify a non-**Boolean** expression, execution stops if the value of the expression has changed since the last time the breakpoint was passed.

ISame as IDD_BP_DLG_CONDITION.

Enter the location (line number, memory address, function, or label) where you want to set the breakpoint. Just to the right of this field is a dropdown list. This list contains the current location (source line number or memory address) for your program. Select this location to enter the current location into the Break At field. Select the Advanced... item in this list to open the Advanced Breakpoints dialog.

Click to display the Breakpoint Condition dialog, where you can enter a conditional expression. The debugger then only stops at the breakpoint if the condition is met.

The breakpoint location or expression. This field is the same as the the Break at field on the Location tab of the Breakpoints dialog. When you click OK, the Location or Data tab will reflect any changes you make here.

Enter the name of the function where the expression or location exists.

Enter the name of the source file where the expression or location exists. If the source file is not in the current directory, include the drive and directory path as well.

Enter the name of the executable file or DLL where the expression or location exists. If the executable file or DLL is not in the current directory, include the drive and directory path.

A variable or expression that determines whether the debugger stops at the breakpoint. The expression can be a **Boolean** expression, which evaluates to true or false, or a non-**Boolean** expression such as a+b/c. If you specify a **Boolean** expression, the execution stops at the breakpoint if the condition evaluates to true. If you specify a non-**Boolean** expression, execution stops if the value of the expression has changed since the last time the breakpoint was reached.

The number of elements that you want to monitor. If you specified a dereferenced pointer, such as *lptr, in the box above, use this field to enter the length (in bytes) of the memory addressed by the pointer. The length must be a positive number.

The number of times the debugger should skip this breakpoint. Set this value if you want to stop when the is met for the Nth time or, if no condition is set, the Nth time the breakpoint is reached. You cannot set this option if the condition evaluates to a non-**Boolean** value.

Check this box if you always want to debug this executable even if it contains no debug symbols (for example, a release build).

Lists the running processes.

Specifies whether the Process List should include processes, for example, RPC services, that were started and controlled by the system.

Displays values in hexadecimal format and parses user input in hexadecimal format in all debugger windows and dialogs. When using the debugger in hexadecimal mode, it is possible to enter decimal numbers using the prefix `0n` (zero-n), for example: `0n1000`.

Displays source code within the listing of assembly-language code.

IDDP_DEBUGIDC_DEBUG_CODEBYTESDisplays bytes corresponding to each assembly-language instruction.

Displays symbolic names (such as **CMyApp::OnMyEvent**) for addresses in the Disassembly Window.

Displays values passed to each parameter for each call shown in the Call Stack window .

Displays type information for each parameter for each call shown in the Call Stack window.

Displays function return values in the Variables window.

The beginning address for the block of memory to be displayed.

Choose the display format for memory contents.

Dynamically evaluates an expression entered in the Memory window. Select this option if you want to enter a pointer, for example, and have the memory window display the address pointed to even when the pointer changes. Do not select this option if you want the pointer to be evaluated once and the Memory window to continue to point to that address even when the pointer changes

Displays data as raw bytes as well as in the chosen format.

Displays memory contents in a fixed-width format when checkbox is selected. Use the textbox to specify the width. Width units are determined by the format selected above. For example, if you choose short and set the width to 4, each row in the Memory window will display four short values.

Use this box to specify the row width when Fixed Width is selected. Width units are determined by the format selected above. For example, if you choose short and set the width to 4, each row in the Memory window will display four short values.

Displays Unicode-format strings for debugging international programs.

Displays contents of floating-point registers in the Registers window.

Enables an application launched from the desktop to call the debugger when an error occurs.

Enables debugging of remote procedure calls.

When this option is selected, Edit and Continue applies code changes automatically when you choose a Step, Go, or Run command. Otherwise, code changes are applied only when you choose Apply Code Changes.

Enables the debugger to load COFF-format debugging information, or DLL Exports when debugging information is not available. Selecting this option may affect debugger performance when loading the application to be debugged.

This dialog appears shows all stack frames for which Edit and Continue could not apply code changes.

A list of function calls for which Edit and Continue could not apply code changes, with function name and thread ID. The call stack index indicates how far the call is from the top of the call stack. (The top position is number zero).

The reason why Edit and Continue could not apply code changes for the selected function call.

The reason why Edit and Continue could not apply code changes for the selected function call.

Select to set a temporary breakpoint in code where Edit and Continue could not apply code changes. When the temporary breakpoint is reached, execution halts and Edit and Continue tries to update the code again.

This dialog shows all DLLs loaded by your application. The list is initially sorted according to the order in which the DLLs are loaded. Use the buttons at the top of the list to sort the DLLs by name, memory address, path, or load order.

This list shows all DLLs loaded by your application. The list is initially sorted according to the order in which the DLLs are loaded. Use the buttons at the top of the list to sort the DLLs by name, memory address, path, or load order.

AppHelp topics start here

Program Variable Properties

This properties page appears when you select a variable or expression in the Variables window or Watch window, then select Properties from the Edit window. It provides the following information for the selected variable or expression:

Type
Data type of the variable or expression.

Expression
Name or representation of the variable or expression.

Value
Contents of the variable or expression.

Provides a space for you to type or select the name of the bookmark.

Lists names of active bookmarks. Selecting one of the names allows you to either delete or go to that bookmark.

Displays the path and filename for a selected bookmark. If nothing is displayed, the bookmark has been defined in a new file.

Displays the line number for a selected bookmark.

Click to add the bookmark displayed in the Name box to the list of active bookmarks. This button is disabled if the bookmark already exists or if the Name box is empty.

Click to delete the selected bookmark from the list of active bookmarks. This button is disabled until you select a bookmark.

Click to go to the selected bookmark. This button is disabled until you select a bookmark.

Provides a space for you to type the name of the symbol whose relationship you want to display. If you type a pattern using the * wildcard, symbols matching that pattern are displayed. If you are querying on a file, you must include a file extension. You can use the * wildcard to search for a pattern in the name or extension.

Lists the types of symbol relationships you can display. You can choose from:

Definitions and References
Indicates where symbols are defined and referenced.

File Outline
Displays all user-defined functions, classes, data, macros, and types.

Base Classes and Members
Displays all classes from which the selected class inherits attributes.

Derived Classes and Members
Displays all classes that inherit attributes from the selected class.

Call Graph
Displays relationships among all the functions that the selected function calls.

Callers Graph
Displays relationships among all the functions that call the selected function.

Select this checkbox to make the results of your query case sensitive. Clear this checkbox to make the results of your query case insensitive.

Provides a space for you to specify the search text or the regular expression to match. You can use the menu button to the right of the drop-down list to display a list of regular search expressions. When you select an expression in this list, the expression is substituted as search text in the Find What text box. If you use regular expressions, be sure the Regular Expression option is checked. You can also use the drop-down list to select from a list of up to 16 previous search strings. For more information on regular expressions, search on “regular expressions” in the Index.

Provides a space for you to enter or select the type of file you want to search. The filename extension determines the file type.

Provides a space for you to enter or select the primary folder that you want to search. You can use the browse button (…) to display the Choose Directory dialog box to change drives and directories.

Click to display a list of regular search expressions. When you select an expression in this list, the expression is substituted as search text in the Find What text box. If you use regular expressions, be sure the Regular Expression option is checked.

Click to display the Choose Directory dialog box to change drives and directories.

Select this checkbox to find only the text strings that exactly match the case of the characters in the Find What string. Otherwise, the command finds strings with either uppercase or lowercase characters that match the characters in the Find What string.

Select this checkbox to use regular expressions in the Find What text box.

Select this checkbox to extend the search to folders containing source files for the project.

Select this checkbox to extend the search to folders containing include files for the project.

Select this checkbox to extend the search to subfolders.

When selected, sends the output of the Find in Files command to a second pane. This allows you to view the results of two Find in Files operations at the same time.

Click to select additional folders to search. This button displays an extended dialog that allows you to add, delete, and organize folders for searching.

Provides a space for you to add additional folders for searching. To add a folder to this list, double-click the empty selection. Then type the path and filename or use the browse button (…) to display the Choose Directory dialog box to change drives and directories. To remove a folder from this list, select and delete the text entry.

Provides a space for you to enter or select the text printed at the top of every printed page. The header prints 0.25 from the top of the page. Click the button to the right to display a list of items you can include in a header and a list of alignment options.

Click to display a list of items you can include in the header and a list of alignment options. Select an item to insert its corresponding code, or select an alignment option to change how the header text aligns on the page. The default alignment is centered.

Provides a space for you to enter or select the text that you want printed at the bottom of every page. The footer prints 0.5 from the bottom of the page. Click the button to the right to display a list of items you can include in a footer and a list of alignment options.

Click to display a list of items you can include in the footer and a list of alignment options. Select an item to insert its corresponding code, or select an alignment option to change how the footer text aligns on the page. The default alignment is centered.

Provides a space for you to specify the width, in inches, of the left margin on every printed page.

Provides a space for you to specify the width, in inches, of the right margin on every printed page.

Provides a space for you to specify the width, in inches, of the top margin on every printed page.

Provides a space for you to specify the width, in inches, of the bottom margin on every printed page.

Lists the installed printers. Select the printer you want.

Prints the entire source file.

Prints the selected text. This option is disabled when no text is selected.

Click this to open the Document Properties dialog box. This dialog box allows you to define advanced printing options.

Provides a space for you to specify the search text or the regular expression to match. You can use the menu button to the right of the drop-down list to display a list of regular search expressions. When you select an expression in this list, the expression is substituted as search text in the Find What text box. If you use regular expressions, be sure the Regular Expression option is checked. You can also use the drop-down list to select from a list of up to 16 previous search strings.

Provides a space for you to specify the string of characters to replace the characters found. You cannot use regular expressions in this string, but you can use tagged expressions.

Click to display a list of regular search expressions. When you select an expression in this list, the expression is substituted as search text in the Find What text box. If you use regular expressions, be sure the Regular Expression option is checked. You can also use the drop-down list to select from a list of up to 16 previous search strings.

Click to display a list of up to nine tagged expressions that you can use as replacement text in the Replace With text box.

Select this checkbox to match text strings only if they are preceded and followed by a space, tab, or punctuation character. Selecting this checkbox also matches text strings if they are at the start or end of a line. Otherwise, the command matches any string, whether it is a fragment of a larger string or not.

Select this checkbox to find only text strings that match the case of the characters in the Find What string exactly. Otherwise, the command finds strings with either uppercase or lowercase characters that match the characters in the Find What string.

Select this checkbox to use regular expressions in the Find What text box.

Replaces strings in the current selection only. This option is disabled when no text is selected.

Replaces strings in the entire file.

Click to repeat the most recent find operation.

Click to replace the currently selected string with the string specified in the Replace With text box.

Click to replace all strings that match the Find What string automatically, without requiring confirmation for each replacement.

Lists the instances in which an identical symbol occurs. Choose one of the instances to examine.

Select this checkbox to include vertical scroll bars in editor windows.

Select this checkbox to allow reuse of a single document window when the process of debugging, browsing, or traversing the list of build errors would otherwise create multiple windows.

Select this checkbox to enable drag-and-drop text editing so you can move or copy selected text with the mouse.

Select this checkbox to include horizontal scroll bars in editor windows.

Select this checkbox to display a margin to the left of each line of text. You can use this margin to select text and display information about source lines, including breakpoints, instruction points, and tag pointers.

Select this checkbox to specify that Microsoft Visual C++ save files before you build a project or run a build utility such as NMAKE.

Select this checkbox to specify that Microsoft Visual C++ automatically reload externally modified files that have been loaded (but not yet changed) by the editor.

Select this checkbox if you want Microsoft Visual C++ to display a prompt so that you can confirm that you want to save a file.

When selected, the editor displays a list of valid member functions and variables when you type "." or "->" after a variable name. When cleared, you can still invoke the Members list with a key combination. The default key combination is CTRL+ALT+T but you can assign different keys by choosing Customize from the Tools menu, selecting the keyboard tab and then selecting the ListMembers entry under the Edit category for the Text editor.

When selected, the editor displays comments for the highlighted item in the Members list (the popup window that appears to help you complete your statement). Displays all comments that use the // or /* delimiters.) This option does not have a keyboard or menu equivalent.

When selected, enables you to view type information for an identifier by simply hovering the mouse over the identifier. When cleared, you can still invoke the type information with a key combination. The default key combination is CTRL+T but you can assign different keys by choosing Customize from the Tools menu, selecting the Keyboard tab and then selecting the TypeInfo entry under the Edit category for the Text editor.

When selected, the editor displays the complete prototype for a function, including any required parameters, after you type the opening parenthesis "(". When cleared, you can still invoke the parameter information with a key combination. The default key combination is CTRL+SHIFT+SPACE but you can assign different keys by choosing Customize from the Tools menu, selecting the Keyboard tab and then selecting the ParameterInfo entry under the Edit category for the Text editor.

Use this text box in addition to the ‘Enable C++Doc comments’ check box, above, to view only those comments that contain the string you specify here. Do not include the comment delimiter (// or /*) in the tag. Enter multiple tags by separating them with a comma (a comma cannot be part of a tag). For example, if you enter [MyTag] in this field, and then tag certain comments in your code (“// [MyTag] *Comment text starts here*…”), only the tagged comments will be displayed when you enable C++Doc comments.

Invisible control

Lists available editor emulations. Select the editor emulation you want from the standard Visual C++ text editor (Visual C++ 4.0 and later), the Visual C++ 2.0 editor, or the BRIEF® or Epsilon™ editors. When you choose an editor from this list, the text editor emulates the key bindings, text selection characteristics, and most editing commands of the selected editor. The default editor is Visual C++ Developer Studio.

Displays the compatibility options and their settings for the selected editor emulation. You can modify the Text editor's behavior by selecting the following options:

Disable Backspace At Start Of Line

Prevents joining of lines by using the BACKSPACE key.

Enable Copy Without Selection

Enables the copy command to copy the entire line in which the cursor is placed.

Enable Line-mode Pastes

Pastes previously copied lines into the text above the current line (and not at the cursor position).

Enable Virtual Space

Allows the cursor into locations that do not currently contain text.

Include Caret Positioning In Undo Buffer

Retraces previous caret positions using the Undo command.

Indent Separate Paragraphs

Paragraphs normally are defined as lines of text between white space. When you are using paragraph commands such as ParaDown and ParaUp, this option treats any line beginning with a tab character as a new paragraph.

Protect read-only files from editing

Prevents you from using the Text editor to modify read-only files.

Use BRIEF's Regular Expression Syntax

Applies BRIEF regular expression syntax to your selected Text editor. If you have selected BRIEF emulation, then BRIEF regular expression syntax is available already.

Double-click in dialog editor edits code (MFC only)

Switches to the Text editor and displays the code associated with a control or a dialog box if you double-click.

Lists the available file types that the editor recognizes. Select the file type you want. Your choice determines the default tab and indent settings. The file type of Default specifies all files not otherwise identified, and is equivalent to plain text.

Provides a place for you to specify the number of space characters that equal one tab character. The default is four space characters.

Provides a space for you to specify the column width in spaces between indent locations. The default is four spaces. Pressing the TAB key once moves the cursor to the next indent location.

Tab Size and Indent Size work together in this way: If you have an Indent Size of seven characters, a Tab Size of three characters, and the insertion point in a text file is at an indent location, pressing the TAB key once inserts two tab characters (each three spaces wide) plus one space character (one space wide) to move the seven character positions to the next indent location.

Converts tabs to the number of spaces specified in the Tab Size box when you save a file.

Saves tabs as tab characters when you save a file.

Does not indent source code automatically.

Indents source code using the default tab and indent sizes.

Indents source code using the Smart Indent options: Indent open brace, Indent closing brace, and Previous lines used for context. If you do not specify any Smart Indent options, Microsoft Visual C++ uses the previous 100 lines for context and indents your source code to match.

Select this checkbox to indent opening braces by the Indent size value.

Select this checkbox to indent closing braces by the Indent size value.

Provides a space for you to specify how many lines to scan back to determine a context for indents. The default value is 100.

AppHelp topics start here

Source File Properties: General

File name
The name of the current source file.
Language
The programming language associated with the file extension. You can select a different language by clicking the one you want on the drop-down list.
Size
The size of the file in both bytes and lines.
Saved
The time and date the file was last saved.
Tab size
The number of space characters that equals one tab character. You can change the size by typing a different number in the box.
Indent size
The number of space characters used to indent a line. You can change the indent size by typing a different number in the box.

When selected, indicates that the project being added is a subproject of another project.

Select the name of the project that will contain, as a subproject, the project you are adding.

Opens a project that is part of a source-code control database.

Enter the new file name here.

Add an optional comment about the files.

Lists files to which the specified action can be applied. The list may contain only the files selected in FileView when the dialog box was opened, or all the files to which the specified action can be applied. The set of files listed depends on the Include Only Selected Files In Dialogs option you can set on the Source Control tab in the Options dialog box (Tools menu, Options command).

If the checked items in the list are not the desired files, select or clear the check boxes to change the selection.

When you click OK, the specified action is applied to the selected files. For example, if you are adding files to source control, the list contains files currently in your project, but not currently under source-code control. When you click OK, the selected files are placed under source-code control.

Click this to select all the files.

When selected, checks the files out to you after adding the master versions of the files to the source-code control project.

Displays differences between your local copy and the master copy.

#Click to access more options. If your source-code control system does not support these options, this button is inactive.

When selected, displays a dialog box that enables you to add files to source-code control whenever you add them to a project.

When selected, displays a dialog box that enables you to copy the latest versions of the master files under source-code control to your workspace when you open the project workspace.

When selected, displays a dialog box that enables you to check files in if they are checked out when you close the workspace.

When selected, displays a dialog box that enables you to check out a file from source-code control if you attempt to edit the file while it is checked in.

Enter the user’s login name here.

When selected, updates the status of the files under source-code control at the most convenient times, as a background task.

When selected, prompts the user with a dialog box when files are being checked out. Otherwise, the selected files are checked out immediately, without prompting the user for confirmation.

When selected, prompts the user to add each new project to source control when it is created.

When selected, only the files selected in FileView are listed in the source control dialog boxes, such as Add to Source Control and Check In File(s). Otherwise, the list contains all the files to which the specified action can be applied.

Displays the name of the source-code control project to which the working directory will apply.

Note This is not a project in your Visual C++ project workspace, it is a project in your source-code control database.

Displays a suggested working directory. To change the working directory, type the desired path or click Browse and select it.

If you change the suggested working directory, you should choose the new path carefully. The definition of the working directory and the best choice for its location depend on the source-code control software you are using. For example, if you are using Visual SourceSafe, the root working directory for a source-code control project should be at or above the level of all the files you intend to add to the source-code control project. Suppose your project contains include files or header files that do not reside within or below the folder containing the project (.dsp) file. The related source-code control project should have a working directory that contains all of these files within its nested subdirectories.

Note for Visual SourceSafe users: The working directory you specify will be the working folder in Visual SourceSafe.

Opens a dialog box where you can browse and select a directory.

App Help topics start here

Properties: General

This read-only property page displays information about the currently selected project when it has been unloaded from the workspace. In workspaces containing multiple projects, unloading projects when they are not needed may improve performance.

If you want to build the project or make any changes in it, you must load the project first. To load the project, select it in the FileView pane, right-click to display the shortcut menu, and click **Load Project**. To unload the project again later, select it in the FileView pane, right-click to display the shortcut menu, and click **Unload Project**.

Project Name
Displays the name of the project.

Project Filename
Displays the complete path and filename of the project (.dsp) file.

Properties: General

This read-only property page displays information about the current workspace displayed in the FileView pane.

Workspace Name
Displays the name of the workspace.

Workspace Path
Displays the complete path to the workspace (.dsw) file.

Status
Shows the source-code control status if the file is under source-code control in a source-code control system that conforms to the Microsoft Source Code Control API Specification.

Specifies the editor that Visual C++ uses to open a file. If you specify “Auto,” Visual C++ determines the editor based on the filename extension or the contents of the file.

Displays a list of Active X controls you can insert into a dialog in the Dialog editor. Inserting a control from this dialog does not generate a wrapper class. If you need a wrapper class, you may use ClassWizard to create one. If an Active X control does not appear in this dialog, try installing the control according to the vendor’s instructions.

Displays the file in which the control is found.

Lists the hue (color wheel value) of the color you are defining. Values range from 0 to 240, where 0 is red, 60 is yellow, 120 is green, 180 is cyan, 200 is magenta, and 240 is blue.

Displays the solid color closest to the dithered color selected on the palette.

Lists the luminosity (brightness) of the color you are defining. Values range from 0 to 240.

Click to scroll and change the luminosity (brightness) of the color you are defining. Values range from 0 to 240.

Click to change the values of a selected color. Position the crosshair on the color you want to change. Then move the slider up or down to change the luminosity or RGB values of the color.

Specifies the red value of the color you are defining. Values range from 0 to 255.

Specifies the saturation value of the color you are defining. Saturation is the amount of color in a specified hue. Values range from 0 to 240.

Species the green value of the color you are defining. Values range from 0 to 255.

Specifies the blue value of the color you are defining. Values range from 0 to 255.

Provides a space for you to enter the width for the toolbar buttons you are converting from a bitmap resource to a toolbar resource. The images are cropped to the width and height specified, and the colors are adjusted to use standard toolbar colors.

Provides a space for you to enter the height for the toolbar buttons you are converting from a bitmap resource to a toolbar resource. The images are cropped to the width and height specified, and the colors are adjusted to use standard toolbar colors.

Provides a space for you to enter the height for the custom image in pixels.

Provides a space for you to enter the width of the custom image in pixels.

Provides a space for you to choose the number of colors for the custom image: 2, 16, or 256.

Provides a space for you to enter the name of a custom resource type. If you do not enter the name in all capital letters, Visual C++ will capitalize the name for you when you exit.

Displays the name of the file to import as a resource.

When enabled, adds rulers to the layout tools; guides can be placed in the rulers. The default guides are the margins, which can be moved by dragging. Click in the rulers to place a guide. Controls “snap” to guides when the controls are moved over or next to them. Controls also move with a guide once they are attached to it. When a control is attached to a guide on each side, and a guide is moved, the control is resized.

Displays the settings for the layout guides.

Displays the settings for the grid spacing in dialog box units (DLUs).

Hides layout tools.

Creates a layout grid. New controls will automatically align to the grid.

Sets the height of the layout grid in dialog box units (DLUs). A vertical DLU is the average height of the dialog box font divided by eight.

Sets the width of the layout grid in dialog box units (DLUs). A horizontal DLU is the average width of the dialog box font divided by four.

Lists the available icon image types. Select the image type you want to open.

Lists the available icon image types. Select the image type you want to open.

Click to create a new icon image with a custom size and number of colors.

Specifies the height of each tile block. This is useful when drawing bitmaps containing multiple images that are arranged at regular intervals.

Specifies the width of each tile block. This is useful when drawing bitmaps containing multiple images that are arranged at regular intervals.

When checked, displays a grid around each pixel in the image editor. The grid appears only at 4x and higher resolutions.

When selected, displays a grid around blocks of pixels in the image editor, specified by the grid spacing values.

Selects the language for the resource copy.

Lists all the available languages in which you can create resources.

Provides a space for you to enter a defined symbol if you want the resource to be built only when that symbol is defined for your configuration.

Click to import a bitmap, icon, cursor, or sound (.WAV) resource to the current resource file from a file on disk.

Specifies the kind of resource you want to create.

Click to add a new custom resource to the current resource file. Custom resources can be edited in the binary editor only.

Click to create the selected resource and begin editing. The resource opens inside the appropriate editor, in the right-hand pane of the workspace. For example, creating a new dialog resource opens it inside the dialog editor.

Enables you to include header files containing shared (read-only) or calculated symbols. Items in the Read-Only Symbol Directives box or the Compile-Time Directives box are included in the resource file exactly as shown. Make sure what you type does not contain any spelling or syntax errors.

Type the name of the symbols file you want to use with the current .RC script.

Enables you to include resource files that are added to your project at compile time. Make sure there are no spelling or syntax errors.

Click to open the Change Symbol dialog box, which you use to change the name or value of a symbol. If the symbol is for a control or resource in use, the symbol can be changed only from the corresponding resource editor.

Click to delete the selected symbol. Available for unused symbols only.

Click to open the New Symbol dialog box, which you use to define the name and, if necessary, a value for a new symbolic resource identifier.

When checked, displays read-only resources. Normally the Symbol Browser displays only the modifiable resources in your resource script file. Modifiable resources appear in bold text; read-only resources appear in plain text.

Click to open the resource that contains the symbol in the corresponding resource editor.

Shows the name and numeric value for each symbol. When the In Use field is checked, it specifies that the symbol is being used by one or more resources. The resource or resources are listed in the Used by box.

Displays the name of the symbol.

Displays the resource or resources using the symbol selected in the symbols list. To move to the editor for a given resource, select the resource in the Used By box and choose View Use, or double-click the resource.

Displays the name of the symbol. Names may not contain spaces, the first character cannot be a number, and the maximum number of characters is 248.

Displays the numeric value of the symbol.

AppHelp topics start here

Accelerator Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Key

The accelerator key. It can be one of the following:

- Integer: Range 0 to 255. It is interpreted as an ASCII or virtual-key value, depending on the Type property. A single digit is interpreted as a key value. To enter an ASCII value from 0 to 9, precede the number with two zeros (for example, 006).
- Character: Single character optionally preceded by ^ to signify a control character.
- Virtual key identifier: Any one of the virtual-key identifiers in the drop-down list.

Next key typed

When you choose this command, the next key combination typed changes the **Key** and **Modifiers** values appropriately. The key is always interpreted as a virtual key if possible.

Modifiers

Indicates whether the accelerator is a combination formed with CTRL, ALT, or SHIFT. When the key is an ASCII value, Ctrl and Shift are not available. Type: Bool. Defaults: Ctrl is True, Alt and Shift are False.

Type

Specifies whether the **Key** property is an ASCII value or a virtual key (VirtKey) value.

Accelerator Table Properties: Resource

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Language

Contains the language used for this resource.

Condition

Determines the inclusion of the resource. For example, if the condition is _DEBUG, this resource would be included only in debug builds.

ActiveX Control Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Visible

Determines whether the control is visible when the application is first run. Type: Bool. Default: True.

Disabled

Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.

Group

Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.

Tabstop

Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.

Help ID

Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

ActiveX Control Properties: All

To change or edit a value, select the value from the list of values and edit the current value as it appears in the edit box.

Animate Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Visible

Determines whether the control is visible when the application is first run. Type: Bool. Default: True.

Disabled

Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.

Group

Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.

Tabstop

Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.

Help ID

Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Animate Properties: Styles

Center

Centers an AVI clip in the animation control's window and leaves the control size and position unchanged when the AVI clip is opened. If you do not specify this style, the control will be resized to the size of the images in the AVI clip when the clip is opened. Type: Bool. Default: False.

Transparent

Draws the AVI clip with a transparent background instead of the background color specified in the clip. Type: Bool. Default: False.

Autoplay

Starts playing the AVI clip as soon as it is opened. When the clip is done playing, it repeats automatically.

Border

Creates a border around the control. Type: Bool. Default: True.

Animate Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.

Bitmap/Icon Properties: Palette

Shows available palette colors. If you have selected 16-color images, you have sixteen choices; if you have selected 256-color images, you have 256 choices.

Click a color on the **Palette** tab to change the color indicator on the **Colors** palette of the Graphics editor to the selected color.

Double-click a color to display the **Custom Color Selector** dialog box, where you can create a custom color and set the color indicator to that color.

Bitmap Properties: File

File name
The name of the file containing the bitmap resources.

Width
Image width in pixels. Type: Integer. Default: 48.

Height
Image height in pixels. Type: Integer. Default: 48.

Colors
Monochrome (2), 16, or 256. The number of colors in a bitmap is determined by the current display device.

Save compressed
Saves the image in compressed format to save space. This option is useful for large bitmaps. Only color bitmaps can be compressed. Type: Bool. Default: False.

Bitmap Properties: General

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Width
Image width in pixels. Type: Integer. Default: 48.

Height
Image height in pixels. Type: Integer. Default: 48.

Colors
Monochrome (2), 16, or 256. The number of colors in a bitmap is determined by the current display device.

File name
The name of the file containing the bitmap resources.

Save compressed
Saves the image in compressed format to save space. This option is useful for large bitmaps. Only color bitmaps can be compressed. Type: Bool. Default: False.

Bitmap Properties: Resource

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Language
Contains the language used for this resource.
Condition
Determines the inclusion of the resource. For example, if the condition is _DEBUG, this resource would be included only in debug builds.
File name
The name of the file containing the resource.
Preview
A box showing what the bitmap looks like. It is useful for browsing through graphics resources without opening them.

Block Header Properties: General

Language ID

Contains the language used for this project.

Code Page

Contains the code page used for this project.

Check Box Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Caption

The text that labels the control. To make one of the letters in the caption of a control the mnemonic key, precede it with an ampersand (&). Default: A name based on the type of control (for example, “Check”) plus a number based on the resource identifier.

Visible

Determines whether the control is visible when the application is first run. Type: Bool. Default: True.

Disabled

Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.

Group

Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.

Tabstop

Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.

Help ID

Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Check Box Properties: Styles

Auto
Creates a check box that, when selected, automatically toggles between checked and unchecked states. You must set this property to True if you are using a group of check boxes with Dialog Data Exchange. Type: Bool. Default: True.

Left text
Positions the check box's caption text to the left instead of to the right. Type: Bool. Default: False.

Tri-state
Creates a three-state check box. A three-state check box can be grayed as well as checked or not checked. A grayed check box indicates that the state represented by the control is undetermined. Type: Bool. Default: False.

Push-like
Makes a button (such as a check box, three-state check box, or radio button) look and act like a push button. The button looks raised when it isn't pushed or checked, and sunken when it is pushed or checked. Type: Bool. Default: False.

Multi-line
Wraps the button text to multiple lines if the text string is too long to fit on a single line in the button rectangle.Type: Bool. Default: False.

Notify
Notifies the parent window if a check box has been clicked or double-clicked. Type: Bool. Default: True.

Flat
Makes a button look flat, not three-dimensional. Type: Bool. Default: False.

Icon
Specifies that the button displays an icon (BS_ICON) instead of text.Type: Bool. Default: False.
The BS_ICON is a new button style in Windows 95. After a button is created with this style, assign the icon to the button by sending a BM_SETIMAGE message to the button with the wParam as IMAGE_ICON and lParam as a handle to the icon. Windows displays the specified icon on the button. The system handles cleanup of icons or cursors loaded from resources. Therefore, the icon should not be deleted unless the icon was created at run time by using **CreateIcon** function.

Bitmap
Specifies that the button displays a bitmap (BS_BITMAP) instead of text.Type: Bool. Default: False.
The BS_BITMAP is a new button style in Windows 95. After a button is created with this style, assign the bitmap to the button by sending a BM_SETIMAGE message to the button with the wParam as IMAGE_BITMAP and lParam as a handle to the bitmap. Windows displays the specified bitmap on the button. The attached bitmap should not be deleted until the button is destroyed.

Horizontal alignment
Positions the control's caption text to the left, center, right or default position in the control.

Vertical alignment
Positions the control's caption text to the top, bottom, center or default position in the control.

Check Box Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.
Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.
Right-to-left reading order
The dialog box text is displayed in right-to-left reading order for languages such as Hebrew or Arabic. Type: Bool. Default: False.

Combo Box and Extended Combo Box Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Visible

Determines whether the control is visible when the application is first run. Type: Bool. Default: True.

Disabled

Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.

Group

Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.

Tabstop

Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.

Help ID

Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Combo Box and Extended Combo Box Properties: Data

Enter listbox items

Allows you to populate a listbox with data. Type the first item and press CTRL+RETURN to add another item.

Combo Box and Extended Combo Box Properties: Styles

Type (applies to Combo Box and Extended Combo Box)

Specifies the type of combo box. This property can have one of the following values:

- Simple: Creates a simple combo box that combines an edit-box control (which takes user input) with a list control. The list is displayed at all times, and the current selection in the list is displayed in the edit-box control.
- Dropdown (Default): Creates a drop-down combo box. This type is the same as a simple combo box, except the list is not displayed unless the user clicks a drop-down arrow at the right of the edit-box control portion of the combo box.
- Drop List: This type is the similar to the drop-down style, but the edit-box control is replaced by a static-text item (which does not take user input) that displays the current selection in the list.

Owner draw (Combo Box)

Controls the owner-draw characteristics of the combo box. Always Fixed for extended combo box control. Other possible values for standard combo box control are as follows:

- No (default for standard combo box control): Turns off the owner-draw style. The combo box contains strings.
- Fixed: Specifies that the owner of the combo box is responsible for drawing its contents and that the items in the combo box are the same height.

  CWnd::OnMeasureItem is called when the combo box is created and **CWnd::OnDrawItem** is called when a visual aspect of the combo box has changed.
- Variable: Specifies that the owner of the combo box is responsible for drawing its contents and that the items in the combo box are variable in height.

  CWnd::OnMeasureItem is called for each item in the list when the combo box is created and **CWnd::OnDrawItem** is called when a visual aspect of the combo box has changed.

Has strings (Combo Box)

Specifies that an owner-draw combo box contains items consisting of strings. The combo box maintains the memory and pointers for the strings so the application can use the LB_GETTEXT message to retrieve the text for a particular item. By default, all combo boxes except owner-draw combo boxes have this style. An application can create an owner-draw combo box either with or without this style.
This style is available only if the **Owner draw** property is set to either Fixed or Variable. If **Owner draw** is set to No, the combo box contains strings by default. Type: Bool. Default: False.

Sort (Combo Box)

Sorts the contents of the combo box alphabetically. Type: Bool. Default: True.

Vertical scroll (Combo Box)

Creates a combo box with a vertical scroll bar. Type: Bool. Default: True.

No integral height (Combo Box)

Specifies that the size of the combo box is exactly the size specified by the application when it creates the combo box. Normally, Windows sizes a combo box so that the combo box does not display partial items. Type: Bool. Default: False.

OEM convert (Combo Box)

Converts text entered in the combo-box control from the Windows character set to the OEM character set and then back to the Windows set. This ensures proper character conversion when the application calls the **AnsiToOem** function to convert a Windows string in the edit-box control to OEM characters. This style is most useful for combo-box controls that contain filenames. Type: Bool. Default: False.

Auto Hscroll (Combo Box)

Automatically scrolls text to the right when the user types a character at the end of the line. Type: Bool. Default: False.

Disable no scroll (Combo Box)

Shows a disabled vertical scroll bar in the combo box when the box does not contain enough items to scroll. Without this style, the scroll bar is hidden when the combo box does not contain enough items to scroll. Type: Bool. Default: False.

Uppercase (Combo Box)

Converts all text to uppercase in both the selection field and the list.Type: Bool. Default: False.

Lowercase (Combo Box)

Converts all text to lowercase in both the selection field and the list. Type: Bool. Default: False.

Combo Box and Extended Combo Box Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.
Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.
Right-to-left reading order
The dialog box text is displayed in right-to-left reading order for languages such as Hebrew or Arabic. Type: Bool. Default: False.
Left scrollbar
Vertical scroll bar (if present) is to the left of the client area. Type: Bool. Default: False.

Control Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Caption

The text that labels the control. To make one of the letters in the caption of a control the mnemonic key, precede it with an ampersand (&). Default: A name based on the type of control (for example, “Check”) plus a number based on the resource identifier.

Visible

Determines whether the control is visible when the application is first run. Type: Bool. Default: True.

Disabled

Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.

Group

Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.

Tabstop

Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.

Help ID

Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Control Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
No Call OLE Init
Prevents the control from calling the **OleInitialize** function when created. Useful only in dialog templates because **CreateWindowEx** does not accept this style.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.
Control parent
Allows the user to navigate among the child windows of the dialog by using the TAB key.
Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.
Right-to-left reading order
The dialog box text is displayed in right-to-left reading order for languages such as Hebrew or Arabic. Type: Bool. Default: False.
Left scrollbar
Vertical scroll bar (if present) is to the left of the client area. Type: Bool. Default: False.

Cursor Properties: File

File name

The name of the file containing the cursor resource.

Width

Image width in pixels. This property is determined by the target-device definition selected by the user. (This property is displayed only and cannot be modified on the property page).

Height

Image height in pixels. This property is determined by the target-device definition selected by the user. (This property is displayed only and cannot be modified on the property page).

Hot spot

Location of the cursor's active area. This property is specified in pixels, relative to the upper-left corner (0,0). It is set with the Set Hot Spot button on the toolbar of the Graphics editor. (This property is displayed only and cannot be modified on the property page).

Cursor Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Width

Image width in pixels. This property is determined by the target-device definition selected by the user. (This property is displayed only and cannot be modified on the property page).

Height

Image height in pixels. This property is determined by the target-device definition selected by the user. (This property is displayed only and cannot be modified on the property page).

Hot spot

Location of the cursor's active area. This property is specified in pixels, relative to the upper-left corner (0,0). It is set with the Set Hot Spot button on the toolbar of the Graphics editor. (This property is displayed only and cannot be modified on the property page).

File name

The name of the file containing the cursor resource.

Custom Control Properties: General

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Visible
Determines whether the control is visible when the application is first run. Type: Bool. Default: True.
Disabled
Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.
Help ID
Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.
Group
Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.
Tabstop
Specifies that the user can move to this control with the TAB key. Type: Bool. Default: True.
Caption
The text that labels the control. To make one of the letters in the caption of a control the mnemonic key, precede it with an ampersand (&). Default: A name based on the type of control (for example, “Check”) plus a number based on the resource identifier.
Class
The name of the control’s Windows class. This class must be registered before the dialog box containing the control is created.
Style
A 32-bit hexadecimal value specifying the control’s style, primarily used to edit the lower 16 bits that make up a user control’s sub-style.
ExStyle
A 32-bit hexadecimal value specifying the control’s extended style.

Custom Resource Properties: File

File name

The name of the file containing the resource.

Custom Resource Properties: Resource

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Language

Contains the language used for this resource.

Condition

Determines the inclusion of the resource. For example, if the condition is _DEBUG, this resource would be included only in debug builds.

File name

The name of the file containing the resource (if the resource is external).

External file

If selected, the resource contents are stored in an external file listed in File Name. Type: Bool. Default: True.

Date and Time Picker Properties: Styles

Format

One of the following:

- Short Date: Displays the date in “4/19/96” format. Default.
- Long Date: Displays the date in “Friday, April 19, 1996” format.
- Time: Instead of the date, displays the time, in “5:13:42 PM” format.

Right Align

Specifies that the calendar drop-down aligns to the right of the drop-down control. When not true, the calendar aligns to the left. Type: Bool. Default: True.

Use Spin Control

Creates the control with up and down arrows on the right side. The user clicks the arrows to modify date values, instead of using the month calendar drop-down control (which is the default). Type: Bool. Default: False.

Show None

Displays a checkbox in the left-hand side of the control which, when cleared, provides a way to have no date currently selected in the control. You can also set this state with **DTM_SETSYSTEMTIME** wParam = GDT_NONE. Verify the state with **DTM_GETSYSTEMTIME**. Type: Bool. Default: False.

Allow Edit

Allows the owner to parse user input and take necessary action. Enables the user to edit within the client area of the control when they select the date and press the F2 key. The control sends **DTN_USERSTRING** notification messages when the user is finished. This enables the owner of the control to parse user input and take necessary action. Type: Bool. Default: False.

Dialog Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Caption

The text that labels the dialog box. Default: A name based on the type of resource (in this case, “Dialog”) plus a number based on the resource identifier assigned by Visual C++.

Font name

The typeface of the font that will be used in all the controls in the dialog box. The bold version of the typeface is always used. Change this value by choosing the Font command in the lower-left corner of the property page. Default: MS Sans Serif.

Menu

Contains the resource identifier of the menu used in the dialog box, if any. Type: Resource identifier. Default: None.

Font size

The point size of the font that will be used in all the controls in the dialog box. Default: 8 points. Change this value by choosing the Font command in the lower-left corner of the property page.

Font

Choose the Font command to change the typeface or size of the dialog-box font.

X Pos

The x-coordinate, in dialog box units (DLUs), of the upper-left corner of the dialog box. Type: Integer.

Y Pos

The y-coordinate, in DLUs, of the upper-left corner of the dialog box. Type: Integer.

Class name

Identifier of a registered dialog class (a Windows operating system window class, not to be confused with a C++ class). This identifier is provided to support C programming. If you are using a resource file with Microsoft Foundation Class Library support, this option is disabled.

To enable this option, switch to ResourceView and display properties for the top-level node (for example, Script1). Clear the **Enable MFC Features** checkbox. Now when you display properties for a dialog, this option is enabled. Type: Integer or String. String must be in double quotes. Default: None.

Dialog Properties: Styles

Style

One of the following:

- Overlapped: Creates an overlapped window. An overlapped window is always a top-level window and should have a caption and a border.
- Popup (Default): Creates a pop-up window.
- Child: Creates a child window.

Border

One of the following:

- None: No border. A title bar is not available.
- Thin : A thin border.
- Resizing: Creates a thick border that can be used to resize the dialog box.
- Dialog Frame (Default): A dialog-box border.

Titlebar

Creates a title bar for the dialog box. This check box is cleared if the dialog box has no border. Type: Bool. Default: True.

System menu

Creates a system menu for the dialog box. This check box is disabled if there is no title bar. Type: Bool. Default: True.

Minimize box

Creates a minimize box for the dialog box. This check box is disabled if there is no title bar. Type: Bool. Default: False.

Maximize box

Creates a maximize box for the dialog box. This check box is disabled if there is no title bar. Type: Bool. Default: False.

Clip siblings

Clips child windows relative to each other; that is, when a particular child window is repainted, this style clips all other top-level child windows out of the region of the child window to be updated. If **Clip siblings** is False and child windows overlap, it is possible, when drawing in the client area of a child window, to draw in the client area of a neighboring child window. **Clip siblings** is for use with child windows only. Type: Bool. Default: False.

Clip children

Excludes the area occupied by child windows when drawing within the parent window. This option is used when creating the parent window. Do not use this style if your dialog box contains a group box. Type: Bool. Default: False.

Horizontal scroll

Creates a horizontal scroll bar for the dialog box. Type: Bool. Default: False.

Vertical scroll

Creates a vertical scroll bar for the dialog box.

If you create horizontal or vertical scroll bars for a dialog box that uses the default border style (the Dialog Frame style), the scroll bars are drawn overlapping the borders of the dialog rather than within them, and the contents of the dialog box are clipped improperly. This is standard Windows behavior. To avoid this behavior, use a different border style when creating a scrollable dialog box. Type: Bool. Default: False.

Dialog Properties: More Styles

System modal
Creates a system-modal dialog box, which prohibits switching to another window or program while the dialog box is active. Type: Bool. Default: False.

Absolute align
Determines whether the dialog box is aligned relative to the screen or relative to its parent window. If **Absolute align** is True, the dialog is displayed at coordinates relative to the upper-left corner of the screen. Type: Bool. Default: False.

Visible
Specifies that the dialog box is visible when first displayed. Set this property to False for form views and dialog-bar template resources. Type: Bool. Default: True.

Disabled
Creates a dialog box that is initially disabled. Type: Bool. Default: False.

Context help
Includes a question mark in the title bar of the window. When the user clicks the question mark, the cursor changes to a question mark with a pointer. If the user then clicks a child window, the child receives a WM_HELP message. The child window should pass the message to the parent window procedure, which should call the WinHelp function using the HELP_WM_HELP command. The Help application displays a pop-up window that typically contains help for the child window.

Set Foreground
Brings the dialog box to the foreground. Internally, Windows calls the SetForegroundWindow function for the dialog box. Type: Bool. Default: False.

3D-look
Gives the dialog box a nonbold font and draws three-dimensional borders around control windows in the dialog box. Type: Bool. Default: False.

No fail create
Creates the dialog box even if errors occur — for example, if a child window cannot be created or if the system cannot create a special data segment for an edit control. Type: Bool. Default: False.

No idle message
Suppresses the WM_ENTERIDLE message ordinarily sent to a dialog box's owner when no more messages are waiting in its message queue. Type: Bool. Default: False.

Control
Creates a dialog box that works well as a child window of another dialog box, much like a page in a property sheet. This style allows the user to tab among the control windows of a child dialog box, use its accelerator keys, and so on. Type: Bool. Default: False.

Center
Centers the dialog box in the working area — that is, the area not obscured by the tray. Type: Bool. Default: False.

Center mouse
Centers the mouse cursor in the dialog box. Type: Bool. Default: False.

Local edit
Specifies that edit-box controls in the dialog box will use memory in the application's data segment. Normally, all edit-box controls in dialog boxes use memory outside the application's data segment. Type: Bool. Default: False.

Dialog Properties: Extended Styles

Tool window
Creates a tool window; that is, a window intended to be used as a floating toolbar. A tool window has a title bar that is shorter than a normal title bar, and the window title is drawn using a smaller font. Type: Bool. Default: False.

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.

Static edge
Creates a border around the dialog box. Type: Bool. Default: False.

Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.

Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.

Control parent
Allows the user to navigate among the child windows of the dialog by using the TAB key.

Context help
Includes a question mark in the title bar of the window. When the user clicks the question mark, the cursor changes to a question mark with a pointer. If the user then clicks a child window, the child receives a WM_HELP message. The child window should pass the message to the parent window procedure, which should call the WinHelp function using the HELP_WM_HELP command. The Help application displays a pop-up window that typically contains help for the child window.

No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.

Right-to-left reading order
The dialog box text is displayed in right-to-left reading order for languages such as Hebrew or Arabic. Type: Bool. Default: False.

Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.

Left scrollbar
Vertical scroll bar (if present) is to the left of the client area. Type: Bool. Default: False.

Dialog Properties: Resource

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Language
Contains the language used for this resource.
Condition
Determines the inclusion of the resource. For example, if the condition is _DEBUG, this resource would be included only in debug builds.
Preview
A box showing what the dialog looks like.

Dialog Properties: Resource

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Language

Contains the language used for this resource.

Condition

Determines the inclusion of the resource. For example, if the condition is _DEBUG, this resource would be included only in debug builds.

Preview

A box showing what the bitmap looks like. It is useful for browsing through graphics resources without opening them.

Edit Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Visible

Determines whether the control is visible when the application is first run. Type: Bool. Default: True.

Disabled

Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.

Group

Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.

Tabstop

Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.

Help ID

Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Edit Properties: Styles

Align text
Text aligns left, centered, or right when Multiline is selected. Default: Left.

Multi-line
Creates a multiline edit-box control. By default, if a multiline edit-box control has focus and a user presses ENTER, the dialog box responds as if the user had chosen the default command button.
If you want users to be able to insert new lines in multiline edit-box controls by pressing ENTER, you should enable either the **AutoHScroll** or **Want return** properties.
When **AutoHScroll** is selected, the multiline edit-box control automatically scrolls horizontally when the caret goes past the right edge of the control. Pressing ENTER starts a new line.
When **AutoHScroll** is not selected, the multiline edit-box control automatically wraps words to the beginning of the next line when necessary. Pressing ENTER starts a new line only if the **Want return** property is enabled.
The window size determines the wordwrap position. If the window size changes, the wordwrap position also changes, and the text is redisplayed.
Multiline edit-box controls can have scroll bars. An edit-box control with scroll bars processes its own scroll-bar messages. Edit-box controls without scroll bars scroll as described in the previous paragraph. They also process any scroll messages sent by the parent window. Type Bool. Default: False.

Number
Prevents the user from typing non-numeric characters. Type: Bool. Default: False.

Horizontal scroll
Provides a horizontal scroll bar for a multiline control. Type: Bool. Default: False.

Auto HScroll
Automatically scrolls text to the right when the user types a character at the right end of the box. Type: Bool. Default: True.

Vertical scroll
Provides a vertical scroll bar for a multiline edit-box control. Type: Bool. Default: False.

Auto VScroll
In a multiline control, **Auto VScroll** automatically scrolls text up one line when the user presses ENTER on the last line. Type: Bool. Default: False.

Password
Displays all characters as an asterisk (*****) as they are typed into the edit-box control. This property is not available in multiline controls. Type: Bool. Default: False.

No hide selection
Changes the way text is displayed when an edit box loses and regains focus. If **No hide selection** is set to True, selected text in an edit box is displayed as selected at all times. Type: Bool. Default: False.

OEM convert
Converts text typed in the edit-box control from the Windows character set to the OEM character set and then back to the Windows set. This ensures proper character conversion when the application calls the **AnsiToOem** function to convert a Windows string in the edit-box control to OEM characters. This style is most useful for edit-box controls that contain filenames. Type: Bool. Default: False.

Want return
Specifies that a carriage return be inserted when the user presses the ENTER key while typing text into a multiline edit-box control. If this style is not specified, pressing the ENTER key has the same effect as pressing the dialog box's default command button. This style has no effect on a single-line edit-box control. Type: Bool. Default: False

Border
Creates a border around the edit box. Type: Bool. Default: True.

Uppercase
Converts all characters to uppercase as they are typed into the edit box. Type: Bool. Default: False.

Lowercase
Converts all characters to lowercase as they are typed into the edit box. Type: Bool. Default: False.

Read-only
Prevents the user from typing or editing text in the edit box. Type: Bool. Default: False.

Edit Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.
Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.
Right-to-left reading order
The dialog box text is displayed in right-to-left reading order for languages such as Hebrew or Arabic. Type: Bool. Default: False.
Left scrollbar
Vertical scroll bar (if present) is to the left of the client area. Type: Bool. Default: False.

FILEFLAGS Properties: General

VS_FF_DEBUG

Specifies that the file contains debugging information.

VS_FF_PRERELEASE

Specifies that the file is a development version.

Group Box Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Caption

The text that labels the control. To make one of the letters in the caption of a control the mnemonic key, precede it with an ampersand (&). Default: A name based on the type of control (for example, “Check”) plus a number based on the resource identifier.

Visible

Determines whether the control is visible when the application is first run. Type: Bool. Default: True.

Disabled

Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.

Group

Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.

Tabstop

Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.

Help ID

Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Group Box Properties: Styles

Horizontal alignment

Positions the control's caption text to the left, center, right or default position in the control.

Icon

Specifies that the button displays an icon (BS_ICON) instead of text.Type: Bool. Default: False.
The BS_ICON is a new button style in Windows 95. After a button is created with this style, assign the icon to the button by sending a BM_SETIMAGE message to the button with the wParam as IMAGE_ICON and lParam as a handle to the icon. Windows displays the specified icon on the button. The system handles cleanup of icons or cursors loaded from resources. Therefore, the icon should not be deleted unless the icon was created at run time by using **CreateIcon** function.

Bitmap

Specifies that the button displays a bitmap (BS_BITMAP) instead of text.Type: Bool. Default: False.
The BS_BITMAP is a new button style in Windows 95. After a button is created with this style, assign the bitmap to the button by sending a BM_SETIMAGE message to the button with the wParam as IMAGE_BITMAP and lParam as a handle to the bitmap. Windows displays the specified bitmap on the button. The attached bitmap should not be deleted until the button is destroyed.

Notify

Notifies the parent window if a group box has been clicked or double-clicked. Type: Bool. Default: False.

Flat

Makes a button look flat, not three-dimensional. Type: Bool. Default: False.

Group Box Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.

Static edge
Creates a border around the dialog box. Type: Bool. Default: False.

Modal frame
Provides a 3D frame.

Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.

Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.

No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.

Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.

Right-to-left reading order
The dialog box text is displayed in right-to-left reading order for languages such as Hebrew or Arabic. Type: Bool. Default: False.

Hot Key Properties: General

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Visible
Determines whether the control is visible when the application is first run. Type: Bool. Default: True.
Disabled
Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.
Group
Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.
Tabstop
Specifies that the user can move to this control with the TAB key. Type: Bool. Default: True.
Help ID
Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Hot Key Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.

Static edge
Creates a border around the dialog box. Type: Bool. Default: False.

Modal frame
Provides a 3D frame.

Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.

Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.

No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.

Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.

Right-to-left reading order
The dialog box text is displayed in right-to-left reading order for languages such as Hebrew or Arabic. Type: Bool. Default: False.

Icon Properties: File

File name

The name of the file containing the icon resource.

Width

Image width in pixels. This property is determined by the currently selected target-device definition. (This property is displayed only and cannot be modified on the property page).

Height

Image height in pixels. This property is determined by the currently selected target-device definition. (This property is displayed only and cannot be modified on the property page).

Colors

Monochrome (2) or 16. This property is determined by the currently selected target-device definition. (This property is displayed only and cannot be modified on the property page).

Icon Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Width

Image width in pixels. This property is determined by the currently selected target-device definition. (This property is displayed only and cannot be modified on the property page).

Height

Image height in pixels. This property is determined by the currently selected target-device definition. (This property is displayed only and cannot be modified on the property page).

Colors

Monochrome (2) or 16. This property is determined by the currently selected target-device definition. (This property is displayed only and cannot be modified on the property page).

File name

The name of the file containing the icon resource.

IP Address Properties

There are no editable properties unique to the IP Address control.

List Box Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Visible

Determines whether the control is visible when the application is first run. Type: Bool. Default: True.

Disabled

Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.

Group

Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.

Tabstop

Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.

Help ID

Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

List Box Properties: Styles

Selection

Determines how items in a list box can be selected. Possible values are as follows:

- Single (default): Users can select only one item in a list box at a time.
- Multiple: Users can select more than one list-box item at a time, but cannot extend the selection from where it starts. The SHIFT and CTRL keys can be used together with the mouse to select and deselect items, including non-adjacent items. Clicking or double-clicking an unselected item selects it. Clicking or double-clicking a selected item deselects it.
- Extended: Users can extend the selection by dragging. The SHIFT and CTRL keys can be used together with the mouse to select and deselect list-box items, select groups of items, and select non-adjacent items.
- None: Users cannot select any items.

Owner draw

Controls the owner-draw characteristics of the list box. Possible values are as follows:

- No (default): Turns off the owner-draw style. The list box contains strings.
- Fixed: Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are the same height.

  CWnd::OnMeasureItem is called when the list box is created and **CWnd::OnDrawItem** is called when a visual aspect of the list box has changed.
- Variable: Specifies that the owner of the list box is responsible for drawing its contents and that the items in the list box are variable in height.

  CWnd::OnMeasureItem is called for each item in the list when the list box is created and **CWnd::OnDrawItem** is called when a visual aspect of the list box has changed.

Has strings

Specifies that an owner-draw list box contains items consisting of strings. The list box maintains the memory and pointers for the strings so the application can use the LB_GETTEXT message to retrieve the text for a particular item. By default, all list boxes except owner-draw list boxes have this style. An application can create an owner-draw list box either with or without this style.
This style is available only if the **Owner draw** property is set to either Fixed or Variable. If **Owner draw** is set to No, the list box contains strings by default. Type: Bool. Default: False.

Border

Creates a border around the list box. Type: Bool. Default: True.

Sort

Sorts the contents of the list box alphabetically. Type: Bool. Default: True.

Notify

Notifies the parent window if a list item has been clicked or double-clicked. Type: Bool. Default: True.

Multi-column

Specifies a multicolumn list box that is scrolled horizontally. The LB_SETCOLUMNWIDTH message sets the width of the columns. Type: Bool. Default: False.

Horizontal scroll

Creates a list box with a horizontal scroll bar. Type: Bool. Default: False.

Vertical scroll

Creates a list box with a vertical scroll bar. Type: Bool. Default: True.

No redraw

Specifies that the list box's appearance is not updated when changes are made. You can change this style at any time by sending a WM_SETREDRAW message or by calling **CWnd::SetRedraw**. Type: Bool. Default: False.

Use tabstops

Allows a list box to recognize and expand tab characters when drawing its strings. The default tab positions are 32 dialog box units (DLUs). Type: Bool. Default: False.

Want key input

Specifies that the owner of the list box receives WM_VKEYTOITEM or WM_CHARTOITEM messages whenever the user presses a key and the list box has the input focus. This allows an application to

perform special processing on the keyboard input. If a list box uses the **Has Strings** style, the list box receives WM_VKEYTOITEM messages. If a list box does not use the **Has Strings** style, it receives WM_CHARTOITEM messages. Type: Bool. Default: False.

Disable no scroll

Shows a disabled vertical scroll bar in the list box when the box does not contain enough items to scroll. Without this style, the scroll bar is hidden when the list box does not contain enough items to scroll. Type: Bool. Default: False.

No integral height

Specifies that the size of the list box is exactly the size specified by the application when it created the list box. Normally, Windows sizes a list box so that the list box does not display partial items. Type: Bool. Default: True.

List Box Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.
Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.
Right-to-left reading order
The dialog box text is displayed in right-to-left reading order for languages such as Hebrew or Arabic. Type: Bool. Default: False.
Left scrollbar
Vertical scroll bar (if present) is to the left of the client area. Type: Bool. Default: False.

List Control Properties: General

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Visible
Determines whether the control is visible when the application is first run. Type: Bool. Default: True.
Disabled
Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.
Group
Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.
Tabstop
Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.
Help ID
Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

List Control Properties: Styles

View

Sets the display view for the list control:

- Icon (default): Specifies icon view.
- Small Icon: Specifies small icon view.
- List: Specifies list view.
- Report: Specifies report view.

Align

Aligns the icons in the list:

- Top (default): Top aligns the icons in the view.
- Left: Left aligns the icons in the view.

Sort

Sorts the icons in the list in the following order:

- None (default): No sort applied.
- Ascending: Sorts items based on item text in ascending order.
- Descending: Sorts items based on item text in descending order.

Single selection

Allows only one item at a time to be selected. By default, multiple items may be selected. Type: Bool. Default: False.

Auto arrange

Specifies that icons are automatically kept arranged in icon and small icon view. Type: Bool. Default: False.

No label wrap

Displays item text on a single line in icon view. By default, item text may wrap in icon view. Type: Bool. Default: False.

Edit labels

Allows item text to be edited in place. The parent window must process the LVN_ENDLABELEDIT notification message. Type: Bool. Default: False.

No scroll

Disables scrolling. All items must be within the client area. Type: Bool. Default: True.

No column header

Specifies that a column header is not displayed in report view. By default, columns have headers in report view. Type: Bool. Default: False.

No sort header

Specifies that column headers do not work like buttons. This style is useful if clicking a column header in report view does not carry out an action, such as sorting. Type: Bool. Default: False.

Show selection always

Uses the system highlight colors to draw the selected item. Type: Bool. Default: False.

List Control Properties: More Styles

Owner draw fixed

Specifies that the parent window draws the tabs in the control. Type: Bool. Default: False.

Share image list

Specifies that the control does not take ownership of the image list assigned to it; that is, the control does not destroy the image list when it is destroyed. This style enables the same image lists to be used with multiple controls. Type: Bool. Default: False.

Border

Creates a border around the list box. Type: Bool. Default: True.

Owner data

Specifies a "virtual" list view control. A "virtual" list view is a list view control that has the LVS_OWNERDATA style. Because this type of list control is intended for large data sets, it is recommended that you cache requested item data to improve retrieval performance. Type: Bool. Default: False.

List Control Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.
Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.
Left scrollbar
Vertical scroll bar (if present) is to the left of the client area. Type: Bool. Default: False.

Menu Item Properties: General

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Caption
The text that labels the menu item. To make one of the letters in the caption of a menu item the mnemonic key, precede it with an ampersand (&).

Separator
If True, the menu item is a separator. Type: Bool. Default: False.

Checked
If True, the menu item is initially checked. Type: Bool. Default: False.

Pop-up
If True, the menu item is a pop-up menu (a submenu). Type: Bool. Default: True for top-level menu items on a menu bar; otherwise False.

Grayed
If True, the menu item is initially grayed and inactive. Type: Bool. Default: False.

Inactive
If the **Grayed** property is True, then the **Inactive** property is always True. Otherwise **Inactive** determines whether the menu item is initially inactive. Type: Bool. Default: False.

Help
Right-justifies the menu item on the menu bar at run time. Type: Bool. Default: False.

Break
Can be one of these values:

- None (Default): No break.
- Column: For static menu-bar items, this value places the item on a new line. For pop-up menus, this value places the item in a new column with no dividing line between the columns. Setting this property affects the appearance of the menu only at run time, not in the menu editor.
- Bar: Same as Column except, for pop-up menus, this value separates the new column from the old column with a vertical line. Setting this property affects the appearance of the menu only at run time, not in the menu editor.

Prompt
Contains text to appear in the status bar when this menu item is highlighted. The text is placed in the string table with the same identifier as the menu item. This property is available only in resource files with Microsoft Foundation Class Library (MFC) support.

Menu Item Properties: Extended Styles

Right-to-left order and alignment

Allows menu items to display right to left when the interface is localized to any language that reads right-to-left, such as Hebrew or Arabic.

Menu Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Menu Properties: Resource

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Language

Contains the language used for this resource.

Condition

Determines the inclusion of the resource. For example, if the condition is _DEBUG, this resource would be included only in debug builds.

Preview

A box showing what the bitmap looks like. It is useful for browsing through graphics resources without opening them.

Month Calendar Properties: Styles

Day States

Enables you to specify which date is displayed in bold, instead of having the control automatically use the system date. You must handle the **MCN_GETDAYSTATE** message. Type: Bool. Default: False.

Multi Select

Enables the user to select a range of dates rather than just one day at a time. By default, the maximum range is one week. You can change the maximum selectable range by using the MCM_SETMAXSELCOUNT message. Type: Bool. Default: False.

No Today Circle

See "No Today," below. This style removes the circle but leaves today's date selected. Type: Bool. Default: False.

No Today

By default, the Calendar control displays today’s date in bold at the bottom of the control, and with a circle around the actual date. Set this style if you do not want to display a “Today” selection. Type: Bool. Default: True.

Week Numbers

Use this style to have the control display week numbers (1 through 52) to the left of each row of week days. Type: Bool. Default: False.

Picture Properties: General

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Visible
Determines whether the control is visible when the application is first run. Type: Bool. Default: True.
Disabled
Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.
Help ID
Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.
Group
Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.
Tabstop
Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.
Type
Sets the type of static graphic to display one of the following:

- Frame (Default): Displays a frame. You set the color of the frame in the **Color** box. Use a frame to visually group controls.
- Rectangle: Displays a filled rectangle. You set the color of the rectangle in the **Color** box.
- Icon: Displays an icon in the dialog box. Use the image box to specify the identifier of the icon you want to display.
- Bitmap: Displays a bitmap in the dialog box. Use the image box to specify the identifier of the bitmap you want to display.
- Enhanced Metafile: Displays an enhanced metafile in the dialog box.

Image
Select the identifier of the icon or bitmap to display. This property is available only when the picture type is icon or bitmap.
Color
Sets the color of a frame or rectangle to black, white, gray, or etched. Etched gives it a 3-D appearance. This property is not available when the picture type is icon, bitmap or enhanced metafile. Default: Black.

Picture Properties: Styles

Sunken
Creates a border with a sunken edge around the picture control. Type: Bool. Default: False.

Border
Creates a border around the picture. Type: Bool. Default: True.

Notify
Notifies the parent window if a picture has been clicked or double-clicked. Type: Bool. Default: False.

Center image
Specifies that, if the bitmap or icon is smaller than the client area of the picture control, the rest of the client area is filled with the color of the pixel in the top left corner of the bitmap or icon. Type: Bool. Default: False.

Right justify
Specifies that the lower right corner of a picture control is to remain fixed when the control is resized. Only the top and left sides are adjusted to accommodate a new bitmap or icon. Type: Bool. Default: False.

Real size image
Prevents a static icon or bitmap control from being resized as it is loaded or drawn. If the icon or bitmap is larger than the destination area, the image is clipped. Type: Bool. Default: False.

Picture Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.

Progress Properties: General

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Visible
Determines whether the control is visible when the application is first run. Type: Bool. Default: True.
Disabled
Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.
Group
Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.
Tabstop
Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.
Help ID
Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Progress Properties: Styles

Border
Creates a border around the control. Type: Bool. Default: True.
Vertical
Displays and fills the progress control vertically, instead of horizontally. Type: Bool. Default: False.
Smooth
Fills the progress control “smoothly,” rather than using a segmented fill. Type: Bool. Default: False.

Progress Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.

Static edge
Creates a border around the dialog box. Type: Bool. Default: False.

Modal frame
Provides a 3D frame.

Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.

Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.

No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.

Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.

Push Button Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Caption

The text that labels the control. To make one of the letters in the caption of a control the mnemonic key, precede it with an ampersand (&). Default: A name based on the type of control (for example, “Check”) plus a number based on the resource identifier.

Visible

Determines whether the control is visible when the application is first run. Type: Bool. Default: True.

Disabled

Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.

Group

Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.

Tabstop

Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.

Help ID

Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Push Button Properties: Styles

Default button

If True, the control is the default button in the dialog box. The default button is drawn with a heavy black border when the dialog box first appears and is executed if the user presses ENTER without choosing another command in the dialog box. Windows allows only one default button in a dialog box. Type: Bool. Default: False.

Owner draw

Creates an owner-draw button. Use an owner-draw button when you need to customize the appearance of a control by providing your own **OnDrawItem** message handler in the owner-window procedure (usually a dialog-box procedure or class derived from the Microsoft Foundation class **CDialog** or **CFormView**). You can also derive your own class from **CButton** and override **CButton::DrawItem**. See **CWnd::OnDrawItem** and **CButton::OnDraw** in the *Class Library Reference* for more information.

Icon

Specifies that the button displays an icon (BS_ICON) instead of text.Type: Bool. Default: False.
The BS_ICON is a new button style in Windows 95. After a button is created with this style, assign the icon to the button by sending a BM_SETIMAGE message to the button with the wParam as IMAGE_ICON and lParam as a handle to the icon. Windows displays the specified icon on the button. The system handles cleanup of icons or cursors loaded from resources. Therefore, the icon should not be deleted unless the icon was created at run time by using **CreateIcon** function.

Bitmap

Specifies that the button displays a bitmap (BS_BITMAP) instead of text.Type: Bool. Default: False.
The BS_BITMAP is a new button style in Windows 95. After a button is created with this style, assign the bitmap to the button by sending a BM_SETIMAGE message to the button with the wParam as IMAGE_BITMAP and lParam as a handle to the bitmap. Windows displays the specified bitmap on the button. The attached bitmap should not be deleted until the button is destroyed.

Multi-line

Wraps the button text to multiple lines if the text string is too long to fit on a single line in the button rectangle.Type: Bool. Default: False.

Notify

Notifies the parent window if a push button has been clicked or double-clicked. Type: Bool. Default: True.

Flat

Makes a button look flat, not three-dimensional. Type: Bool. Default: False.

Horizontal alignment

Positions the control’s caption text to the left, center, right or default position in the control.

Vertical alignment

Positions the control’s caption text to the top, bottom, center or default position in the control.

Push Button Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.
Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.
Right-to-left reading order
The dialog box text is displayed in right-to-left reading order for languages such as Hebrew or Arabic. Type: Bool. Default: False.

Properties: Resource

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Language
Contains the language used for this resource.
Condition
Determines the inclusion of the resource. For example, if the condition is _DEBUG, this resource would be included only in debug builds.
File name
The name of the file containing the resource.
Preview
A box showing what the bitmap looks like. It is useful for browsing through graphics resources without opening them.
External file
If selected, the resource contents are stored in an external file listed in File Name. Type: Bool. Default: True.

Radio Button Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Caption

The text that labels the control. To make one of the letters in the caption of a control the mnemonic key, precede it with an ampersand (&). Default: A name based on the type of control (for example, “Check”) plus a number based on the resource identifier.

Visible

Determines whether the control is visible when the application is first run. Type: Bool. Default: True.

Disabled

Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.

Group

Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.

Tabstop

Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.

Help ID

Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Radio Button Properties: Styles

Auto
When the user selects a radio button with this property, the radio button is automatically selected and any other radio buttons in the same group are cleared (deselected). You must set this property to True if you are using a group of radio buttons with Dialog Data Exchange. Type: Bool. Default: True.

Left text
Places the radio button's caption text on the left rather than the right. Type: Bool. Default: False.

Push-like
Makes a button (such as a check box, three-state check box, or radio button) look and act like a push button. The button looks raised when it isn't pushed or checked, and sunken when it is pushed or checked. Type: Bool. Default: False.

Multi-line
Wraps the button text to multiple lines if the text string is too long to fit on a single line in the button rectangle.Type: Bool. Default: False.

Notify
Notifies the parent window if a radio button has been clicked or double-clicked. Type: Bool. Default: True.

Flat
Makes a button look flat, not three-dimensional. Type: Bool. Default: False.

Icon
Specifies that the button displays an icon (BS_ICON) instead of text.Type: Bool. Default: False.
The BS_ICON is a new button style in Windows 95. After a button is created with this style, assign the icon to the button by sending a BM_SETIMAGE message to the button with the wParam as IMAGE_ICON and lParam as a handle to the icon. Windows displays the specified icon on the button. The system handles cleanup of icons or cursors loaded from resources. Therefore, the icon should not be deleted unless the icon was created at run time by using **CreateIcon** function.

Bitmap
Specifies that the button displays a bitmap (BS_BITMAP) instead of text.Type: Bool. Default: False.
The BS_BITMAP is a new button style in Windows 95. After a button is created with this style, assign the bitmap to the button by sending a BM_SETIMAGE message to the button with the wParam as IMAGE_BITMAP and lParam as a handle to the bitmap. Windows displays the specified bitmap on the button. The attached bitmap should not be deleted until the button is destroyed.

Horizontal alignment
Positions the control's caption text to the left, center, right or default position in the control.

Vertical alignment
Positions the control's caption text to the top, bottom, center or default position in the control.

Radio Button Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.
Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.
Right-to-left reading order
The dialog box text is displayed in right-to-left reading order for languages such as Hebrew or Arabic. Type: Bool. Default: False.

Resource File Properties: General

File name

The name of the file. (This property is displayed only and cannot be modified on the property page.)

Enable MFC features

If selected, the file contains MFC features. Type: Bool. Default: True.

Use 3D controls

Display controls using 3-D effects. Type: Bool. Default: True.

Rich Edit Properties: General

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Visible
Determines whether the control is visible when the application is first run. Type: Bool. Default: True.
Disabled
Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.
Group
Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.
Tabstop
Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.
Help ID
Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Rich Edit Properties: Styles

Align text
Text aligns left, centered, or right when Multiline is selected. Default: Left.

Multi-line
Creates a multiline edit-box control. By default, if a multiline edit-box control has focus and a user presses ENTER, the dialog box responds as if the user had chosen the default command button.
If you want users to be able to insert new lines in multiline edit-box controls by pressing ENTER, you should enable either the **AutoHScroll** or **Want return** properties.
When **AutoHScroll** is selected, the multiline edit-box control automatically scrolls horizontally when the caret goes past the right edge of the control. Pressing ENTER starts a new line.
When **AutoHScroll** is not selected, the multiline edit-box control automatically wraps words to the beginning of the next line when necessary. Pressing ENTER starts a new line only if the **Want return** property is enabled.
The window size determines the wordwrap position. If the window size changes, the wordwrap position also changes, and the text is redisplayed.
Multiline edit-box controls can have scroll bars. An edit-box control with scroll bars processes its own scroll-bar messages. Edit-box controls without scroll bars scroll as described in the previous paragraph. They also process any scroll messages sent by the parent window. Type Bool. Default: False.

Number
Prevents the user from typing non-numeric characters. Type: Bool. Default: False.

Horizontal scroll
Provides a horizontal scroll bar for a multiline control. Type: Bool. Default: False.

Auto HScroll
Automatically scrolls text to the right when the user types a character at the right end of the box. Type: Bool. Default: True.

Vertical scroll
Provides a vertical scroll bar for a multiline edit-box control. Type: Bool. Default: False.

Auto VScroll
In a multiline control, **Auto VScroll** automatically scrolls text up one line when the user presses ENTER on the last line. Type: Bool. Default: False.

Password
Displays all characters as an asterisk (*****) as they are typed into the edit-box control. This property is not available in multiline controls. Type: Bool. Default: False.

No hide selection
Changes the way text is displayed when an edit box loses and regains focus. If **No hide selection** is set to True, selected text in an edit box is displayed as selected at all times. Type: Bool. Default: False.

OEM convert
Converts text typed in the edit-box control from the Windows character set to the OEM character set and then back to the Windows set. This ensures proper character conversion when the application calls the **AnsiToOem** function to convert a Windows string in the edit-box control to OEM characters. This style is most useful for edit-box controls that contain filenames. Type: Bool. Default: False.

Want return
Specifies that a carriage return be inserted when the user presses the ENTER key while typing text into a multiline edit-box control. If this style is not specified, pressing the ENTER key has the same effect as pressing the dialog box's default command button. This style has no effect on a single-line edit-box control. Type: Bool. Default: False

Border
Creates a border around the edit box. Type: Bool. Default: True.

Uppercase
Converts all characters to uppercase as they are typed into the edit box. Type: Bool. Default: False.

Lowercase
Converts all characters to lowercase as they are typed into the edit box. Type: Bool. Default: False.

Read-only
Prevents the user from typing or editing text in the edit box. Type: Bool. Default: False.

Rich Edit Properties: More Styles

Disable no scroll

Shows a disabled vertical scroll bar in the list box when the box does not contain enough items to scroll. Without this style, the scroll bar is hidden when the list box does not contain enough items to scroll. Type: Bool. Default: False.

Rich Edit Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
No Call OLE Init
Prevents the control from calling the **OleInitialize** function when created. Useful only in dialog templates because **CreateWindowEx** does not accept this style.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.
Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.
Right-to-left reading order
The dialog box text is displayed in right-to-left reading order for languages such as Hebrew or Arabic. Type: Bool. Default: False.
Left scrollbar
Vertical scroll bar (if present) is to the left of the client area. Type: Bool. Default: False.

Scrollbar Properties: General

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Visible
Determines whether the control is visible when the application is first run. Type: Bool. Default: True.
Disabled
Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.
Group
Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.
Tabstop
Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.
Help ID
Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Scrollbar Properties: Styles

Align

Value can be set as follows:

- None: (Default).
- Top/Left: Aligns the top and left edges of the scrollbar with the top and left edges of the rectangle defined by the parameters of **CreateWindowEx**.
- Right: Aligns the right and bottom edges of the scrollbar with the right and bottom edges of the rectangle defined by the parameters of **CreateWindowEx**.

Slider Properties: General

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Visible
Determines whether the control is visible when the application is first run. Type: Bool. Default: True.
Disabled
Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.
Group
Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.
Tabstop
Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.
Help ID
Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Slider Properties: Styles

Orientation

Displays the trackbar (slider) horizontally (default) or vertically.

Point

Displays tick marks on either or both sides of the trackbar (slider) in the following orientation:

- Both (Default): Displays tick marks on both sides of the trackbar.
- Top/Left: Displays tick marks on one side of the trackbar; on the top of a horizontal trackbar, to the left of a vertical trackbar.
- Bottom/Right: Displays tick marks on the other side of the trackbar, on the bottom of a horizontal trackbar, and to the right of a vertical trackbar.

Tick marks

Specifies the display of tick marks on a trackbar (slider). Type: Bool. Default: False.

Autoticks

Specifies that a tick mark is placed for each increment in the trackbar's (slider's) range of values. These tick marks are created automatically when an application sends the TBM_SETRANGE message. Type: Bool. Default: False.

Enable selection

Specifies that a selection range is displayed (with triangles and a highlighted area) on the trackbar (slider). Type: Bool. Default: False.

Border

Creates a border around the track bar control. Type: Bool. Default: False.

Tool tips

Specifies that a tool tip is created for the slider.

Slider Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.

Spin Properties: General

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Visible
Determines whether the control is visible when the application is first run. Type: Bool. Default: True.
Disabled
Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.
Group
Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.
Tabstop
Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.
Help ID
Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Spin Properties: Styles

Orientation

Displays the spin (up-down) control horizontally or vertically (default).

Alignment

Value can be set as follows:

- Unattached (Default).
- Left: Positions the up-down control next to the left edge of the buddy window. The buddy window is moved to the right and its width decreased to accommodate the width of the up-down control.
- Right: Positions the up-down control next to the right edge of the buddy window. The width of the buddy window is decreased to accommodate the width of the up-down control.

Auto buddy

Automatically selects the previous window in the Z order as the up-down control's buddy window. (Specify Z order by setting the tab order.) The buddy window displays as text the values set by the spin control.Typically, the buddy window is an edit control or a static text control. Type: Bool. Default: False.

Set buddy integer

Causes the up-down control to set the text of the buddy window (using the WM_SETTEXT message) when the position changes. The text consists of the position formatted as a decimal or hexadecimal string. Type: Bool. Default: False.

No thousands

Does not insert a thousands separator between every three decimal digits. Type: Bool. Default: False.

Wrap

Causes the position to "wrap" if it is incremented or decremented beyond the ending or beginning of the range. Type: Bool. Default: False.

Arrow keys

Causes the up-down control to increment and decrement the position when the UP arrow and DOWN arrow keys are pressed. Type: Bool. Default: True.

Spin Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.

String Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Caption

The actual text of the string.

String Table Properties: Resource

For string tables, the ID is always **STRINGTABLE** and cannot be modified.

Language

Contains the language used for this resource.

Tab Control Properties: General

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Visible
Determines whether the control is visible when the application is first run. Type: Bool. Default: True.
Disabled
Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.
Group
Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.
Tabstop
Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.
Help ID
Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Tab Control Properties: Styles

Alignment

Can be one of the following values:

- Right Justify (Default): Width of each tab is increased so that each row of tabs fills the entire width of the tab control.
- Fixed Width: Sizes tabs to the width of the widest label.
- Ragged Right: In multiline tabs, the width of the tabs does not stretch to fill the rows.

Focus

Can be one of the following values:

- Default: Specifies that the user can use the keyboard to give the input focus to a tab in the control.
- On Button Down: Specifies that a tab recieves the input focus when clicked.
- Never: Specifies that a tab never recieves the input focus when clicked.

Buttons

Specifies that the tabs in the control resemble buttons. Tabs in this type of tab control should serve the same function as button controls; that is, clicking a tab should carry out a command instead of displaying a page. Type: Bool. Default: False.

Tool tips

Specifies that a tool tip is created for each tab in the tab control. Type: Bool. Default: False.

Border

Creates a border around the tab control. Type: Bool. Default: False.

Multiline

Displays multiple rows of tabs.Type: Bool. Default: False.

Owner draw fixed

Specifies that the parent window draws the tabs in the control. Type: Bool. Default: False.

Force icon left

Left aligns the icon, leaving the label centered. Type: Bool. Default: False.

Force label left

Left-aligns both the icon and label. Type: Bool. Default: False.

Tab Control Properties: More Styles

Hottrack

Gives the control a visible outline as the pointer passes over it. Also known as hover selection, or track selection. Type: Bool. Default: False.

Bottom

Displays tabs along the bottom edge of the control rather than the top. This value equals TCS_RIGHT. Type: Bool. Default: False.

MultiSelect

Enables the user to select multiple tabs by holding down CTRL when clicking. Must be used in combination with the Buttons style on the Styles tab of the control properties. Type: Bool. Default: False.

Scroll Opposite

Unneeded tabs scroll to the opposite side of the control when a tab is selected. Type: Bool. Default: False.

Vertical

Displays tabs overlapped along the left edge of the control, with tab text displayed vertically. Type: Bool. Default: False.

Tab Control Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. This window does not obscure any windows that are beneath it. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.
Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.
Right-to-left reading order
The dialog box text is displayed in right-to-left reading order for languages such as Hebrew or Arabic. Type: Bool. Default: False.

Text Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Caption

The text that labels the control. To make one of the letters in the caption of a control the mnemonic key, precede it with an ampersand (&). Default: A name based on the type of control (for example, “Check”) plus a number based on the resource identifier.

Visible

Determines whether the control is visible when the application is first run. Type: Bool. Default: True.

Disabled

Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.

Group

Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.

Tabstop

Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.

Help ID

Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Text Properties: Styles

Align text
Controls how text is aligned in the static-text control. The possible values are Left, Center, and Right. Set this option to Left when No Wrap is selected. Default: Left.

Center Vertically
Centers text vertically within the static-text control. Type: Bool. Default: False.

No prefix
Prevents ampersands (&) in the control's text from being interpreted as the mnemonic character. Normally a string containing an ampersand is displayed with the ampersand removed and the next character in the string underlined. The **No Prefix** style is most often used when filenames or other strings that may contain an ampersand need to be displayed.

No wrap
Displays text left-aligned. Tabs are expanded but words are not wrapped. Text that extends past the end of a line is clipped. Type: Bool. Default: False.

Simple
Disables **No Wrap** and **Text Align**. Text in static text controls with this property set does not wrap and is not clipped. In addition, setting this property means that overriding WM_CTLCOLOR in the parent window has no effect on the control. Type: Bool. Default: False.

Notify
Notifies the parent window if a check box has been clicked or double-clicked. Type: Bool. Default: False.

Sunken
Creates a border with a sunken edge around the static text control. Type: Bool. Default: False.

Border
Creates a border around the text control. Type: Bool. Default: False.

Text Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.
Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.
Right-to-left reading order
The dialog box text is displayed in right-to-left reading order for languages such as Hebrew or Arabic. Type: Bool. Default: False.

Toolbar Button Properties: General

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Width

The width of the toolbar button.

Height

The height of the toolbar button.

Prompt

Contains text to appear in the status bar when this menu item is highlighted. The text is placed in the string table with the same identifier as the menu item. This property is available only in resource files with Microsoft Foundation Class Library (MFC) support.

Toolbar Properties: Resource

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Language
Contains the language used for this resource.
Condition
Determines the inclusion of the resource. For example, if the condition is _DEBUG, this resource would be included only in debug builds.
File name
The name of the file containing the resource.
Preview
A box showing what the toolbar looks like. It is useful for browsing through graphics resources without opening them.

Tree Control Properties: General

ID
A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.
Visible
Determines whether the control is visible when the application is first run. Type: Bool. Default: True.
Disabled
Determines if the resource is displayed as disabled when the dialog box is created. Type: Bool. Default: False.
Group
Specifies the first control of a group of controls in which the user can move from one control to the next by using the arrow keys. All controls in the tab order after the first control with the **Group** property set to False belong to the same group. The next control in the tab order with **Group** set to True ends the first group of controls and starts the next group. Type: Bool. Default: False.
Tabstop
Specifies that the user can move to this control with the TAB key. Type: Bool. Default: False.
Help ID
Assigns a help ID to the control. The help ID is based on the resource ID. Type: Bool. Default: False.

Tree Control Properties: Styles

Has buttons
Displays plus (+) and minus (-) buttons next to parent items. The user clicks the buttons to expand or collapse a parent item's list of child items. To include buttons with items at the root of the tree view, **Lines at root** must be set to true. Type: Bool. Default: False.

Has lines
Uses lines to show the hierarchy of items. Type: Bool. Default: False.

Lines at root
Uses lines to link items at the root of the tree view control. This value is ignored if **Has lines** is set to false.

Border
Creates a border around the tree view control. Type: Bool. Default: True.

Edit labels
Allows the user to edit the labels of tree view items. Type: Bool. Default: True.

Disable drag drop
Prevents the tree view control from sending TVN_BEGINDRAG notification messages. Type: Bool. Default: False.

Show selection always
Uses the system highlight colors to draw the selected item. Type: Bool. Default: False.

Tree Control Properties: More Styles

Check Boxes
Enables check boxes as items in the control. Effectively, when set to this style, the control creates and sets a state image list using DrawFrameControl. You must still set and get the item’s check state.

Full Row Select
Enables full-row selection in the tree view. The entire row of the selected item is highlighted and clicking anywhere on an item's row will cause it to be selected.

InfoTip
The tree view control will send the TVN_GETINFOTIP notification to obtain tooltip information. Use this to supply your own description rather than accepting the default Tool tip.

Scroll
Enables both horizontal and vertical scrollbars in the control.

Tool tips
The tree view control provides yellow pop-up tips that display the full name of an item whose name does not fit in the current view.

Track Select
Gives the control a visible outline as the pointer passes over it. Also known as hover selection, or hot tracking.

Single Expand
Causes only the selected node to be expanded in the tree. Changing selections collapses any other node but the selected one.

Tree Control Properties: Extended Styles

Client edge
Creates a border with a sunken edge around the dialog box. Type: Bool. Default: False.
Static edge
Creates a border around the dialog box. Type: Bool. Default: False.
Modal frame
Provides a 3D frame.
Transparent
A window using this style is transparent. Any windows that are beneath this window are not obscured by this window. A window with this style receives WM_PAINT messages only after all sibling windows beneath it have been updated. Type: Bool. Default: False.
Accept files
A dialog box with this style accepts drag-drop files. If a user drops a file on this dialog box, WM_DROPFILES messages will be sent to the control.Type: Bool. Default: False.
No parent notify
Specifies that the child window does not send the WM_PARENTNOTIFY message to its parent window. Type: Bool. Default: False.
Right aligned text
Specifies that text is right-aligned within the dialog box. Type: Bool. Default: False.
Left scrollbar
Vertical scroll bar (if present) is to the left of the client area. Type: Bool. Default: False.

Version Properties: Resource

ID

A symbol defined in the header file. Type: Symbol, Integer, or Quoted String.

Language

Contains the language used for this resource.

Condition

Determines the inclusion of the resource. For example, if the condition is _DEBUG, this resource would be included only in debug builds.

Displays the name of the command or macro corresponding to the toolbar button or menu item you are modifying.

Displays the name that will appear on the toolbar button or menu item. Type any changes to this name you want in the Button Text box and then click OK. Add an ampersand (&) before any letter in the text if you want to specify that the letter is an accelerator key. For example, the letter i is the accelerator key for F&ile.

When selected, specifies that the toolbar button or menu item will contain only an image.

When selected, specifies that the toolbar button or menu item will contain only text.

When selected, specifies that the toolbar button or menu item will contain both an image and text.

Displays the available images for toolbar buttons or menu items, plus the original image. To change the image on the selected button, click an image under Images.

Pastes an image from the clipboard onto the toolbar button or menu item.

Resets the toolbar button or menu item to its original image.

Displays the available images for toolbar buttons or menu items, plus the original image. To change the image on the selected button, click an image under Images.

When selected, indicates that the direction of the search will be from the current cursor position to the end of the file, and then back to the current cursor position.

Click to place a bookmark on all instances of the word or expression in the Find What box.

When selected, searches for text strings that match the case of the characters in the Find What string exactly. Otherwise, the search finds strings with either uppercase or lowercase characters that match the characters in the Find What string.

Begins the find operation specified by the values displayed in this dialog box.

When selected, finds regular expressions for the search specified in the Find What box.

When selected, searches across all documents currently open in the workspace.

Indicates the direction of the search from the current cursor position.

When selected, indicates that the direction of the search will be from the current cursor position to the beginning of the file, and then back to the current cursor position.

Provides a space for you to specify the search text or the regular expression to match. Type the text or expression in the Find What box. Use the drop-down list to select previous search strings. Use the right-arrow button to the right of the drop-down list to display a list of regular search expressions. When you select an expression in this list, the expression is added as search text in the Find What box. If you use regular expressions, be sure the Regular Expression check box is selected.

When selected, finds text strings that match the Find What string and are preceded and followed by a space, tab, or punctuation character, or are at the start or end of a line. Otherwise, the search finds any matching string, whether it is a fragment of a larger string or not.

Displays a list of regular search expressions. When you select an expression in this list, it is added as search text in the Find What box.

Resets the formatting options for a selected category to the default setting.

Lists the background color for the selected element in the Colors list. Click the arrow to display the standard 16 colors or Automatic. Automatic specifies the use of the color for the same element in the Source Window category.

Lists the available fonts. Type the font name in the box or select it from the drop-down list.

Lists the available font sizes. Type the font size in the box or select it from the drop-down list.

Lists the foreground color for the selected element in the Colors list. Click the arrow to display the standard 16 colors or Automatic. Automatic specifies the use of the color for the same element in the Source Window category.

Lists the window elements for which you can specify a foreground or background color. The list of elements displayed changes depending on the window(s) selected in the Category list. The Text element in the Source Windows category sets the default for all the other Text elements in all other windows. The selection of Automatic in the Foreground or Background lists for the Source Windows category uses the system-wide choices of colors in the Control Panel.

Lists the windows for which you can set formatting options. If you select All Windows, the formatting options are applied to all windows in Visual C++. Categories in bold type specify all windows in that category. Categories in normal type specify windows included in the categories in bold type. For instance, you can set options for all the windows related to the debugger, and then set different options for individual types of debugger windows, such as the Calls Window.

Displays an example of the selected format options.

Assigns the shortcut key combination displayed in the Press New Shortcut Key box to the command selected in the Commands box.

Lists commands on the menu selected in the Categories box. Select the command you want to assign to a shortcut key. A brief description of the command appears in the Description box.

Displays the shortcut key combinations currently assigned to the command selected in the Commands box.

Removes the shortcut key combination that is selected in the Current Keys box.

Restores all the shortcut key combinations for the selected editor to their default setting.

Specifies the editor or view to which you want to assign shortcut keys. For example, select Text if you want to assign a shortcut key that will be available when you are working in the Text editor.

Displays the current command assignment for the shortcut key combination entered in the Press New Shortcut Key box.

Provides a space for you to type a new shortcut key combination. To test a new shortcut key combination for the command selected in the Commands box, press the keys you want to assign. If the key combination is currently assigned to another command, the command is displayed in the Currently Assigned To box. To assign the new shortcut key combination, click Assign.

Lists the menus containing the items to which you can assign a shortcut key. The commands for the selected menu appear in the Commands box below.

Displays a description of the selected command.

Specifies additional arguments for the tool each time you start it. For example, you can use $(TargetArgs) to specify the command-line arguments that are passed to the application you are developing. You can also specify additional arguments for each particular instance of the tool.

Opens a dialog box where you can browse and select a file.

When selected, specifies that the window associated with the tool will close when you close the tool.

Displays the path and filename of the tool currently selected in the Menu Contents list.

Displays the path and filename of the tool currently selected in the Menu Contents list.

When selected, specifies that a console application runs without a console window, and that the application's output is displayed in an output window. Each tool uses a separate output window. You can switch among output windows by selecting a tab at the base of the output window.

Displays a list of the argument macros you can include. For example, if you click Current Directory in the list, $(CurDir) is inserted in the Arguments box at the cursor position.

Displays a list of the macros you can include in the directory specification. For example, if you click Current Directory in the list, $(CurDir) is inserted in the Initial Directory box at the cursor position.

Specifies the working directory of the tool.

Specifies the working directory of the tool.

Lists commands you have added to the Tools menu. To add a command, click the New button above the Menu Contents list, type the text for the menu item in the box at the end of the Menu Contents list, and provide the necessary information in the boxes below. Placing an ampersand (&) before a letter in the menu item text makes the letter an accelerator key. To modify a command, select it in the Menu Contents list and change the information specified in the boxes below. To delete a command, select it and click the Delete button above the Menu Contents list. To move a command up or down on the Tools menu, select the command in the Menu Contents list and click one of the arrows above the list.

When selected, displays a dialog box which prompts for command-line arguments each time you run the tool. You can specify additional arguments for each particular instance of the tool.

When selected, restores your document windows to the positions they last occupied in the project workspace.

When selected, displays the clock on the status bar in the selected window.

When selected, automatically loads the workspace you last worked on.

When selected, displays the status bar in the selected window.

Displays a list of Visual C++ windows. When selected, the windows dock along the edges of the environment. Otherwise, they float away from the edges.

When selected, automatically places the most recently used windows at the top of the list on the Window menu. Otherwise, windows are shown in the order that files were loaded.

When selected, the menu bar that displays at the top of Visual C++ is text-based and is compatible with screen-reading devices and programs used by people who are blind or have low vision. This more traditional menu bar is not customizable.

When selected, the recently used files and workspaces are listed on submenus of the File menu. Otherwise, the recently used files and workspaces are listed on the File menu itself.

Specifies the number of windows that will appear on the Window menu. The default is 10 windows.

Specifies the number of filenames for recently used files that will appear on the File menu (or its Recent Files submenu). The default is 4 filenames.

Specifies the number of recently used workspaces that will appear on the File menu (or its Recent Workspaces submenu). The default is 4 workspace names.

Keeps the Go To dialog box open on your desktop. As long as the pushpin is enabled, the Go To dialog box will remain open on top of all other windows.

Displays a list of Go To items such as Address, Bookmark, and Definition. For example, Address jumps to any valid debugger expression in the Memory or Disassembly window. In the Binary editor, Offset jumps to the specified hexadecimal offset in the file.

Provides space for you to type text for additional selection criteria or to select an item from a list. The additional selection criteria you enter depends on the Go To What selection. For example, if you select Bookmark in the Go To What box, you enter a bookmark name.

Jumps directly to the previous item of the type selected in the Go To What list. For example, if Bookmark is selected in the list box, click Previous to jump to the previous bookmark in the current document.

Click “Go To” to jump directly to the item displayed in the Go To What list and the Go To What edit box. For example, if you select Line in the Go To What list and enter 180 in the Go To What edit box, you will jump in the current document to line 180.

Click “Next” to jump directly to the next item of the type selected in the Go To What list. For example, if Bookmark is selected in the list box, click Next to jump to the next bookmark in the current document.

Provides a space for you to type or select the path for the file. You may be prompted to specify the path for a file when Microsoft Visual C++ does not find it. This may happen if a source file has been moved from the project directory before starting a debugging session, or if an executable file is not in the directory specified during Setup.

Specifies the drive for the file.

Specifies the available directories.

Click to open a dialog box in which you can select a network path for the file.

Lists all of the currently open windows. You can activate a window by either double-clicking your selection in the Select Window box, or by selecting a window and then clicking the Activate button. If you select a single window, you can choose Activate, Save, or Close Window. If you select multiple windows, you can choose Save, Close Window, Tile Vertically, or Tile Horizontally.

Saves the currently selected files in the Select Window list.

Activates the currently selected file in the Select Window list.

Closes the window and file of each file currently selected in the Select Window list.

Tiles the open document windows horizontally.

Tiles the open document windows vertically.

When selected, shows a new Tip of the Day every time you start Microsoft Visual C++.

Displays the next Tip of the Day.

Provides space to display tips on using Microsoft Visual Studio products.

When selected, displays on-screen descriptions of toolbar buttons when you float your cursor over the button.

When selected, displays the shortcut key in the ToolTip for a toolbar button (if one is available).

When selected, enlarges the toolbar buttons so that they are easier to see. This feature is useful when giving presentations or for those who are blind or have low vision.

Enables you to create a new custom toolbar.

Displays the name of the custom toolbar selected in the Toolbars list. You can edit the name by typing in the text box.

Lists both standard and custom toolbars. When selected, the toolbar is displayed. Otherwise, the toolbar is hidden. The Menu Bar cannot be hidden, except when you are editing in full-screen mode.

Deletes the selected custom toolbar.

Resets the selected default toolbar to its original settings.

Resets all default toolbars to the original settings.

Provides a space for you to type the name of a new custom toolbar. The name must be distinct from all other toolbar names. You can use mixed case, but each name must be unique without regard to case.

If you want to change this name later, you can edit the name in the Toolbar Name box on the Toolbars tab.

Lists the default menus with toolbar buttons, commands, and menus that you can add to a toolbar or menu. When you select an item in the list, toolbar buttons, commands, or menus appear in the box on the right and can be dragged onto a toolbar or menu.

Shows the buttons or other items associated with the current selection in the Categories list. To place a button, command, macro, or other item on a toolbar, drag the item you want from the box to the right of the Categories box onto a toolbar. To create a new toolbar, drag the item to the location where you want the new toolbar.

Select a category in the Categories list. When you select an item in the list, toolbar buttons, commands, or menus appear in the box on the right and can be dragged onto a toolbar or menu. To place a button, command, macro, or other item on a toolbar, drag the item you want from the box to the right of the Categories box onto a toolbar. To create a new toolbar, drag the item to the location where you want the new toolbar.

When available, provides a description of the item selected in the Categories box.

Enables you to edit an image for the toolbar button.

Select what menu items are displayed while you are customizing a menu. For example, if you select All Editors, all the commands available on the menu are displayed at one time, but only while you are customizing the menu. After you close the Customize dialog box, only commands for the current editor are displayed.

Resets all default menus to their original image, commands, and submenus. Default submenus are restored to their original form and custom submenus are removed. The Menu Bar is also reset.

Click the command or other item you want to add. If the item you want is not shown, select a different category. To display a list of every available command, select the All Commands category.

Click the type of file you want to create. To add the new file to an existing project when it is created, select Add to Project, select the project name, and type a filename in the File Name box. A default location for the new file is supplied automatically, but you can change the drive and path.

To create a new file without adding it to any existing project, select a file type and click OK. A new file of the selected type is opened with a default name. You provide the filename and location when you save the new file.

If you want to create an Active document, such as a spreadsheet, choose the Other Documents tab.

Opens a dialog box where you can select a directory.

When selected, indicates that the new file will be added to an existing project.

When selected, indicates that the new file will be added to an existing project.

Provides a space for you to select the name of the project to which you want to add the file.

Provides a space for you to type the name of the new file. If you do not provide an extension, the default extension for the selected file type is appended automatically.

Provides a space for you to type the name of the new file. If you do not provide an extension, the default extension for the selected file type is appended automatically.

Provides a space for you to specify the directory for the new file. You can type a path or select one by using the browse button to the right of the Location box.

Provides a space for you to specify the directory for the new file. You can type a path or select one by using the browse button to the right of the Location box.

Provides a space for you to specify the folder where you want to place the new file in the project tree.

Click the type of document file you want to create. To add the new file to an existing project when it is created, select Add to Project, select the project name, and type a filename in the File Name box. A default location for the new file is supplied automatically, but you can change the drive and path.

To create a new file without adding it to any existing project, select a file type and click OK. A new file of the selected type is opened with a default name. You provide the filename and location when you save the new file.

The file types listed here are Active documents, created by using COM components such as Microsoft Excel or Microsoft Word. If you want to create a Visual C++ file type, such as a source file, choose the Files tab.

When selected, indicates that the new file will be added to an existing project.

Provides a space for you to select the name of the project to which you want to add the file.

Provides a space for you to type the name of the new document file. If you do not provide an extension, the default extension associated with the document type is appended automatically.

Provides a space for you to type the name of the new document file. If you do not provide an extension, the default extension associated with the document type is appended automatically.

Provides a space for you to specify the directory for the new document file. You can type a path or select one by using the browse button to the right of the Location box.

Provides a space for you to specify the directory for the new document file. You can type a path or select one by using the browse button to the right of the Location box.

Click the type of project you want to create. You must also type a project name in the Project Name box. A default location for the new project is supplied automatically, but you can change the drive and path. To add the new project to the open workspace, select Add to Current Workspace. Otherwise, a new workspace is created automatically to contain the new project. To make the new project a subproject of an existing project, select Dependency Of and specify the project name. Select the appropriate platform or platforms.

Because Visual C++ registers projects by name, you should use a unique name for each project you create. This enables Visual C++ to support workspaces that contain projects that are in different layouts and locations.

When selected, indicates that a new project workspace will be created for this new project. The name and location for the new workspace will be the same as the name and location for the new project.

To create a workspace with a name or location different from the name and location for any of the projects it will contain, choose the Workspace tab and create a blank workspace. Then, with the workspace open, you can create a new project and insert it into the workspace at the same time. On the Projects tab, specify the project name and location and select Add to Current Workspace.

When selected, indicates that the new project will be added to an existing project workspace, which must be the currently open workspace.

When selected, indicates that the new project is a subproject of an existing project.

Provides a space for you to select the existing project to which this new subproject belongs.

Select each platform for which the new project will be built.

Select each platform for which the new project will be built.

Provides a space for you to type the name of the new project.

Because Visual C++ registers projects by name, you should use a unique name for each project you create. This enables Visual C++ to support workspaces that contain projects that are in different layouts and locations.

Provides a space for you to type the name of the new project.

Because Visual C++ registers projects by name, you should use a unique name for each project you create. This enables Visual C++ to support workspaces that contain projects that are in different layouts and locations.

Provides a space for you to specify the directory for the new project files. You can type a path or select one by using the browse button to the right of the Location box.

Provides a space for you to specify the directory for the new project files. You can type a path or select one by using the browse button to the right of the Location box.

To use a file wizard to create a set of related files, click the icon for the wizard you want. Some file wizards require you to add the new files to a project. To add the new files to an existing project when they are created, select Add to Project, select the project name, and type a filename in the File Name box. The appropriate extensions are added to the filenames automatically. A default location is supplied automatically, but you can change the drive and path.

To create a new set of files without adding them to any existing project, select a file wizard and click OK. New files of the appropriate types are opened with a default filename. You provide the filename and location when you save the new files.

When selected, indicates that the new files will be added to an existing project.

Provides a space for you to select the name of the project to which you want to add the files.

Provides a space for you to type the filename of the new files. Note: Any extension you provide will be ignored. The extensions for each file are determined by the selected file wizard.

Provides a space for you to type the filename of the new files. Note: Any extension you provide will be ignored. The extensions for each file are determined by the selected file wizard.

Provides a space for you to specify the directory for the new files. You can type a path or select one by using the browse button to the right of the Location box.

Provides a space for you to specify the directory for the new files. You can type a path or select one by using the browse button to the right of the Location box.

Click to specify the type of workspace you want to create. You must also type a workspace name in the Workspace Name box. A default location for the new workspace is supplied automatically, but you can change the drive and path.

If you want the workspace location to be different from the location of any project it will contain, you must create a blank workspace first and then create the projects. If the workspace name and location can be the same as the name and location of a project, you can create both the project and the workspace in one step. To create both in one step, use the Projects tab to create the project and select the option Create New Workspace. In that case, the workspace name and location match the name and location of the new project.

Provides a space for you to type the filename of the new workspace files. The extensions for each file are provided automatically.

Provides a space for you to type the filename of the new workspace files. The extensions for each file are provided automatically.

Provides a space for you to specify the directory for the new workspace files. You can type a path or select one by using the browse button to the right of the Location box.

Provides a space for you to specify the directory for the new workspace files. You can type a path or select one by using the browse button to the right of the Location box.

Lists the names of projects to which you can add the current file.

Lists the names of projects to which you can add the current file.

Automatically opens the file the same way it was last viewed, unless you select to view it using one of the other options, such as Binary or Resources.

Automatically opens the file the same way it was last viewed, unless you select to view it using one of the other options, such as Binary or Resources.

Opens a project that is part of a source-code control database.

Select the builder you want and click OK.

If options are available, you can set them in the builder interface. For example, if you are inserting two controls, a Data Range Header and a Data Range Footer, you can use the Data Range Builder to insert both controls in your .asp file at once. Select the Data Range Wizard from the list of builders and click OK. Then you can choose options for Data Range Header properties in the Data Range Builder interface.

APP Help Starts Here

No Properties Available

Multiple items are selected, but no properties are common to all the selected items. To view the properties, select one item at a time or select multiple items of the same type.

Tip Select the pushpin at the upper-left corner of the property page. When the property page is pinned, it remains visible and the contents change as your selection changes.

No Properties Available

There is no currently selected item that has properties.

No Properties Available

There is no currently selected item that has properties.

A list of the classes that AppWizard will create in the project. When you select a class in this window, information on that class appears in the text boxes below. If the text boxes are active, you can edit their contents. If they are inactive (grey), you cannot edit their contents. Whether a text box is active or not depends on the class selected.

The name of the class that you have selected in the New Classes list box. If the text box is active, you can edit the contents by typing in the text box, then selecting another class in the class list above.

The name of the header file associated with the selected class. If the text box is active, you can edit the contents by typing in the text box, then selecting another class in the class list above.

The class from which the selected class in the New Classes list box is derived. If the combo box is active, you can select another class to be the base class.

The name of the source code file associated with the selected class. If the text box is this active, you can edit the contents by typing in the text box, then selecting another class in the class list above.

IDD_CONFIRM - no control help needed for this dialog

IDD_PROGRESS - no control help needed for this dialog

Select this option to create a Single Document Interface (SDI) architecture for your program, with a view class based on **CView**. A single document interface, such as Windows Notepad, allows a user to work with just one document at a time. You can change the base class for the view in Step 6 of the Wizard. To create a form-based application, for example, you could use **CFormView** for the view class.

Select this option to create a Multiple Document Interface (MDI) architecture for your program, with a view class based on **CView**. A multiple document interface, such as Microsoft Word for Windows, allows a user to open multiple documents, each with its own window. You can change the base class for the view in Step 6 of the Wizard. To create a form-based application, for example, you could use **CFormView** for the view class.

Select this option to create a dialog-based architecture for your program, with a dialog class based on **CDialog**. A dialog-based program, such as MFC Trace Options, displays a simple dialog box for user input based on a dialog-template resource. It is created using a resource compiler DIALOG statement.

Includes Document/View architecture in your application using the **CDocument** and **CView** base classes (default). Clear this checkbox if you're porting a non-MFC application, for example, or to reduce the size of your compiled executable. An application without document/view architecture will not include MFC support for opening a document from a disk file.

Select the language you want to use for your resources. The drop-down list displays the languages available on your system. If you select a language other than English, the appropriate .DLL for that language must already be installed. The file-naming convention is APPWZ*XXX*.DLL where *XXX* is a three-letter language specifier, for example, APPWZJPN.DLL for Japanese.

Select this option if you do not want your program to have ActiveX (formerly OLE) support. By default, AppWizard creates a program without ActiveX support.

Select this option if you want your program to contain linked and embedded objects.

Select this option if you want your program to have the ability to create and manage compound document objects. Note that mini-servers cannot run stand-alone and only support embedded items.

Select this option if you want your program to have the ability to create and manage compound document objects. Full-servers are able to run stand-alone and support both linked and embedded items.

Select this option if you want your program to be both a container and a server. A container is a program that can incorporate embedded or linked items into its own documents. A server is a program that can create Automation items for use by container programs.

Select this option if you want your program to have the ability to create and manage Active documents. If you select this option, you must specify a filename extension for your Active document server; to do this, select the Advanced button in the next AppWizard step.

Select this option if you want your program to have the ability to contain Active documents within its frame. Active documents may include, for example, Internet Explorer documents, or Office documents such as Microsoft Word files or Excel spreadsheets.

Select this option to serialize your container program's documents using the compound-file format. (Available only for programs with the Doc View architecture; see Step 1 of the Wizard.) Compound-file format stores a document that contains one or more Automation objects to one file and still allows access to the files for the individual objects. This option provides load on demand of, and incremental saves to, the objects' native data.

Select this option if you do not want to serialize your container program's documents using the compound-file format. This option forces the loading of an entire file containing objects into memory. Incremental saves to individual objects are not available. If one object is changed and subsequently saved, all objects in the file are saved.

Selecting Automation makes it possible for your program to manipulate objects implemented in another program, or to expose your program to Automation clients.

Select this option if you want your program to use ActiveX controls. By default, the program supports ActiveX controls. If you do not choose this option and, at a later time, want to insert ActiveX Controls into your project, you must add a call to **AfxEnableControlContainer** in your program's **InitInstance** member function.

Select this option if you do not want the program that AppWizard creates to provide database support. This is the default option.

Select this option if you want the basic level of database support for your program. AppWizard creates a header file (AFXDB.H), and link libraries but does not create any database-specific classes. You can create recordsets and use them to examine and update records.

Select this option if you want the program that AppWizard creates to include database header files, link libraries, a record view and a recordset. (Available only for programs with the Doc View architecture; see Step 1 of the Wizard.) This option includes document support but no serialization support. If you choose to include a database view, you must specify the source of the data. The program will have a **CRecordView**-derived class as its view class. This class is associated with a **CRecordset**-derived class, which AppWizard also creates for you. This option gives you a form-based program in which the record view is used to view and update records via its recordset.

Select this option if you want the program that AppWizard creates to include database header files, link libraries, a record view and a recordset. (Available only for programs with the Doc View architecture; see Step 1 of the Wizard.) This option supports document serialization, which you can use, for example, to update a user profile file. Database programs typically operate on a per-record basis rather than on a per-file basis and so do not need serialization. However, you may have a special use for serialization. If you choose to include a database view, you must specify the source of the data. The program will have a **CRecordView**-derived class as its view class. This class is associated with a **CRecordset**-derived class, which AppWizard also creates for you. This option gives you a form-based program in which the record view is used to view and update records via its recordset.

Select option to open the Database Options dialog box, which you use to specify the database files for your program. This option is available only if you choose to include a database view in your program.

IDC_DATASOURCE_TEXT - no control help needed for this control

IDC_BOGUS_DEFBTN - no control help needed for this control

IDC_NOMACDAO - no control help needed for this control

Provides a standard MFC application architecture (default).

Implements an Explorer-like application using a splitter window where the left pane is a CTreeView and the right pane is a CListView.

Implements the frame of your application as a two-pane window where the left-hand pane displays a selection control, such as a tree-view, and the right hand pane displays the current selection.

Select this option if you want AppWizard to generate and insert comments in the source files that guide you in writing your program. These comments include indicators where you need to add your own code. This option also creates a readme.txt file in your project directory listing and describing each file in the project. This is the default option.

Select this option if you do not want AppWizard to insert comments in the source files it generates.

Select this option to link the MFC library to your program as a shared DLL. Your program makes calls to the MFC library at run time. This reduces the disk and memory requirements of your program if it is composed of multiple executable files that use the MFC library. Both Win32 and MFC programs can call functions in your DLL.

Select this option to link your program to the static MFC library at build time.

The file extension associated with documents created by your program. For example, if your project is named Foo, the file extension could be .foo. (When entering the file extension, do not include the period.) Entering a file extension allows the Explorer to print your program's documents without launching your program when the document icons are dropped on a printer icon.

This ID is used to label your document type in the system registry.

This text box shows the language in which strings are displayed in the edit boxes of the Localized Strings control group.

The dialog title. Type the name of your dialog in the text box. By default the dialog title is the same as the name of the project.

The type of document under which this file can be grouped. This option then becomes available from the “Save as type:” drop-down in the Windows File Save As dialog box. This option is available only if the selected class is derived from class **CDocument**.

By default this box displays the extension you enter in the File Extension box and names the file type after your program. For example, if your project is named Foo, and the file extension is .foo, the filter name is: “Foo Files (*.foo)” Of course, you can edit the text in this box.

The name that appears in the **New** dialog box if there is more than one new document template. If your program is an Automation server, this name is used as the short name of your Automation object.

If your program is an Automation server, this name is used as the long name of your Automation object. It is also used as the file type name in the system registry.

An ID that the Finder uses to locate a document or program in the file system.

A file attribute that the finder uses to determine which program to use to interpret the associated file’s content.

- Duplicate Help ID; same as **IDD_DOCSTRINGS**:HIDC_FILTER

IDD_NEWAPP is obsolete and should be removed via RAID report.

Select this option to link your program statically to the MFC library at build time. Both Win32 and MFC programs can call functions in your DLL.

- Duplicate Help ID; same as **IDD_PROJ_OPTIONS**:HIDCD_PODLL.

Select this option if you want your program to make calls to the MFC library at run time. This reduces the disk and memory requirements of your program if it is composed of multiple executable files that all use the MFC library. Only MFC programs can call functions in your DLL.

- Duplicate Help ID; same as **IDD_OLE_OPTIONS**:HIDCD_AUTOMATION.

Select this option if you want your program to support Windows sockets. Windows sockets allow you to write programs that communicate over TCP/IP networks.

- Duplicate Help ID; same as **IDD_PROJ_OPTIONS**:HIDCD_POVERBOSE.

- Duplicate Help ID; same as **IDD_PROJ_OPTIONS**:HIDC_RADIO2.

Select this option if you want AppWizard to generate code for a message box called the About box that displays the software version of the program. By default, the program has an About box.

Select this option if you want AppWizard to generate a set of help files for context-sensitive help. Help support requires the help compiler. If you do not have the help compiler, you can install it by re-running Setup.

Select this option if you want the program's interface to have three-dimensional shading. By default, programs have three-dimensional shading.

- Duplicate Help ID; same as **IDD_OLE_OPTIONS**:HIDCD_AUTOMATION.

- Duplicate Help ID; same as **IDD_OLE_OPTIONS**:HIDCD_OCX.

Select this option if you want your program to support Windows sockets. Windows sockets allow you to write programs that communicate over TCP/IP networks.

- Duplicate Help ID; same as **IDD_DOCSTRINGS**:HIDC_APP_TITLE

Select this option to add a toolbar to your program. The toolbar contains buttons for creating a new document; opening and saving document files; cutting copying, pasting, or printing text; displaying the About box; and entering Help mode. Enabling this option also adds menu commands to display or hide the toolbar. By default, programs have a docking toolbar.

Select this option to add a status bar to your program. The status bar contains automatic indicators for the keyboard's CAPS LOCK, NUM LOCK, and SCROLL LOCK keys and a message line that displays help strings for menu commands and toolbar buttons. Enabling this option also adds menu commands to display or hide the status bar. By default, programs have a status bar.

Select this option if you want AppWizard to generate the code to handle the print, print setup, and print preview commands by calling member functions in the **CView** class from the MFC library. (Available only for programs with the Doc View architecture; see Step 1 of the Wizard.) AppWizard also adds commands for these functions to the program's menu. By default, programs have printing support.

- Duplicate Help ID; same as **IDD_DLGAPP_OPTIONS**:HIDCD_POHELP.

- Duplicate Help ID; same as **IDD_DLGAPP_OPTIONS**:HIDCD_PO3D.

Select this option if you want a program that creates, manipulates, transfers, and stores mail messages.

- Duplicate Help ID; same as **IDD_DLGAPP_OPTIONS**:HIDCD_POSOCKETS.

Select this option to implement traditional-style toolbars in your application, which appear across the top of the window, and can be docked or undocked.

Select this option to implement Office-style command bars in your application. Command bars appear similar to standard menus, but can be docked or undocked.

Select this option to implement IE.4-style rebars in your application. Rebars can contain any standard Windows control in addition to menu commands.

Specifies the number of files to be listed on the most recently used list. The default number is 4.

Opens the Advanced Options dialog box, which you use to specify options for document template strings and frame characteristics.

Select this option to enable your program’s windows to use a splitter bar. The splitter bar will split the program's main views. In an MDI program, the MDI child frame's client window is a splitter window, and in an SDI program, the main frame's client window is a splitter window.

Select this option to create a window that has a sizing border. This is the default.

Select this option if you want the main frame window to include a minimize box. This is the default.

Select this option if you want the main frame window to include a maximize box. This is the default.

Specifies that the main frame window include a system menu. This is the default.

Select this option if you want the main frame window to open as an icon.

Select this option if you want the main frame window to open to the full size of the display.

Lists the MDI child frame styles. Select the appropriate options for your program.

Select this option if you want the frame of all MDI child windows to have a sizing border. This is the default.

Select this option if you want MDI child windows to include a minimize box. This is the default.

Select this option if you want MDI child windows to include a maximize box. This is the default.

Select this option if you want MDI child windows to open as icons.

Select this option if you want MDI child windows to open maximized.

Navigate to the previous dialog.

Complete the process now, saving selections you made in previous dialogs and using default settings for the rest of the dialogs.

Lists the name used by Automation clients to request an event from the class. Enter a name or select one from the list.

Lists the name of the member function that sends the event.

Specifies that a stock event, such as a button click, will be implemented for this class. Stock events are defined in the Microsoft Foundation Class (MFC) Library. Available only for Automation-enabled controls, once an External Name has been selected from the list of available stock events.

Lists the parameters for the events of the class. For events you are defining, you may add multiple parameters. If you define a new External Name, you can add parameters in the Parameter List, by double-clicking the first empty row under the Name or Type label. Either enter a parameter or select a type from the list.

Specifies that you will provide your own implementation of an event for this class.

Specifies the name of the Windows message handler function you want to add to the class. When you double-click a dialog control that has no handlers defined, displays the suggested name for the default event of that control. You can change the name.

Lists the name of the method. Automation clients use this name to make requests from the class. Enter a name or select one from the list.

Specifies that a stock method, such as a button click, will be implemented for this method. Stock events are defined in the Microsoft Foundation Class (MFC) Library. Available only for Automation-enabled controls, once an External Name has been selected from the list of available stock events.

Lists the name of the member function that sends the method. Enter an Internal Name if you want it to be distinct from the External Name.

Specifies the return type for the method. Enter a name or select one from the list.

Displays the type of implementation for this class.

Specifies that you will provide your own implementation of an event for the method of this class.

Lists the parameters for the methods or properties of the class. For methods you are defining, you may add multiple parameters. If you define a new External Name, you can add parameters in the Parameter List, by double-clicking the first empty row under the Name or Type label. Either enter a parameter or select a type from the list.

Lists the name of the property. Automation clients use this name to change the property. Enter a property name or select one from the list of predefined properties.

Specifies the return type of the property. Enter a property type or select one from the list.

Lists the C++ class data member variable name associated with the property. If using the Member Variable implementation method, enter the appropriate name.

The name of the user-written function called whenever the property value is changed. Available only when you select the Member Variable implementation method.

Specifies that a stock event, such as a button click, will be implemented for this property. Stock events are defined in the Microsoft Foundation Class (MFC) Library. Available only for Automation-enabled controls, once an External Name has been selected from the list of available stock events.

Specifies that direct access to the member variable will be enabled. Use this option for properties that do not affect the user interface when changed. Use the Notification function if you want to know when the property value is changed.

Specifies that controlled access to the property will be enabled. Use this option when you need to know when the value changes or when a property is calculated. Commonly used for properties that will affect the user interface when changed.

Lists the Get function. If using the Get/Set methods implementation option, enter the appropriate Get method.

Lists the Set function. If using the Get/Set methods implementation option, enter the appropriate Set method.

Lists the parameters for the properties of the class. You may add multiple parameters. To edit the existing parameters in the list, double-click the first empty row under the Name or Type label and either enter a parameter or select a type from the list.

The name of the new variable. By default, ClassWizard provides the m_ prefix to identify it as a member variable.

Specifies that this is a Value variable or a Control variable. For standard Windows controls, choose Value to create a variable that contains the control's text or status as entered by the user. If you choose this option, the variable data will be added to the data type you select in the Variable Type box. Choose Control to create a Control variable that gives you access to the control itself.

Lists available types of variables.

Lists available types of ActiveX properties.

Specifies that for ActiveX controls, the property cannot be changed by the user.

Describes the purpose and function of the member variable.

Lists the name of the header (.H) file in which the class will be declared. Enter a new path and filename or click Browse to select one.

Click to locate the appropriate path and file.

Lists the name of the C++ (.CPP) file in which the class will be defined. Enter a new path and filename or click Browse to select one.

Click to locate the appropriate path and file.

Lists the name of the current project. Select a project name from the list.

Lists the current class. Select a class name from the list.

Click to add a class to your project. Choose from the drop-down menu to create a new class, import a class from a file, or import a class from an ActiveX type library.

Displays the names, source, base class, and resource files for the class.

Displays the name of the .H or .CPP file in which the class will be defined.

Displays the name of the source (implementation) file for the class.

Displays the name of the class from which the selected class is derived.

Displays the name of the resource associated with the selected class, if it is a dialog class.

Displays the message filter, foreign class and foreign variable options for the class.

Lists the message filter currently in effect for the class you have selected. Available filters are categorized by the type of window or dialog box they pertain to. Select a filter name from the list.

Lists the name of the foreign class associated with the class you have selected. The value is “<None>” unless the selected class is a dialog, form view, or record view class. Select a class name from the list.

Lists any foreign variables associated with the class you have selected. This option is generally used for database classes. When you edit a RecordView, you can select the associated **Recordset** class. Type a new variable name.

Lists the classes that will be generated from the type library file. If the class has already been generated, click OK to update the class.

Lists the name of the dispatch class that you want to import.

Lists the name of the .H file that contains header information for the imported dispatch class. Enter a new path and filename or click Browse to select one.

Lists the name of the file to contain implementation code for the imported dispatch class. Enter a new path and filename or click Browse to select one.

Displays the base class from which the dispatch class is inherited.

Displays class information options.

Lists the name of the class you want to add to your program.

Displays the name of the implementation file associated with the class.

Click to change the class file names, or to specify existing class files in which you want to include your new class.

Lists the base class from which the new class will be derived. Select a class from the list.

Lists a dialog ID or other resource ID for the class when one is required. Select a resource ID from the list.

Specifies Automation options for the class. Available only for Automation-capable classes, that is, those derived from **CCmdTarget**.

Specifies that no Automation will be provided for the class.

Specifies that the class enables objects which can be accessed by Automation clients, such as Microsoft Visual Basic and Microsoft Excel.

Specifies that the class will allow Automation clients to create Automation-capable objects directly. To change the name supplied by default, enter the new name.

When the Creatable By Type ID option is enabled, specifies the name that Automation clients will use to request an object of the class. Enter a name.

When enabled, specifies that a class will be created. The class will be attached to the resource.

When enabled, specifies that an existing class will be imported into the ClassWizard database to be attached to the resource. Allows you to reuse a class that you created in a different project.

When enabled, specifies that an existing class in the ClassWizard database will be attached to the resource.

When enabled, specifies that this is a bindable property. This sends an **OnChanged** notification to the data source after this property is changed.

When enabled, specifies that the selected control sends an **OnRequestEdit** notification before the property is changed.

When enabled, specifies that the container should display this property to the end-user as a bindable property.

When enabled, specifies that this property is the default bindable property.

Lists the name of the class whose member variables are displayed for editing. Since you can add variables from this tab, the class list is not filtered for only those classes which have a data map.

Displays the IDs for controls that can be mapped to member variables. The Type column displays the type for the member variable. The Member column displays the name of the member variable. This information is displayed only when a dialog, form view, or record view class is selected.

Describes the properties of the currently selected variable.

Describes the contents of the edit control.

Lists the first validation field. Available only when the currently selected member variable supports validation.

Describes the contents of the edit control.

Lists the second validation field. Available only when the currently selected member variable supports validation. This option is typically used to restrict the legal values of the selected member variable.

Click to add a member variable for the selected control.

Click to delete the selected member variable. Delete Variable is only available if one or more of the items in Control IDs/Column Names is already associated with a member variable.

Click to select a data source containing the table associated with your recordset. After you select a data source, then enable the Tables option to select a table to use. Update Columns is only available for recordset classes.

Click to bind all unbound recordset field data members to the corresponding columns in a table on the data source. By default, the MFC AppWizard and ClassWizard bind all columns, so you seldom need to use Data Binding. If you have unbound some or all columns (by deleting their associated data members), you can rebind them all with Data Binding. Data Binding is only available for recordset classes.

Displays the selected data source.

Displays the tables contained in the selected data source. Select the tables for which you want recordsets created.

Lists the files that will be used to rebuild the .clw file for the project. Initially displays all files from the current project. To rebuild using this default selection, click OK. This list is updated dynamically to reflect any files you add. To add files singly, use the File name: box and click Add. To add a group of files, navigate to the project or folder location and click Add All. Only valid C++ source or header files will be used when rebuilding the .clw file.

Adds the file displayed in the File name: box to the Files in Project list box. The Files in Project list box specifies the files that will be used to rebuild the .clw file for the project.

Adds all files displayed under the File name: box to the Files in Project list box. To rebuild the .clw file with only the files from the current project (default selection) there is no need to add them; simply click OK.

Removes the selected file from the Files in Project list box. Any file you remove will not be included when rebuilding the .clw file.

This control does not appear at run time.

Filters the types of files shown in the File name: list box. You can add only valid C++ source or header files to the ClassWizard database. Select All Files if, for example, you have a C++ source file with a non-standard extension that you want to add. If you select a non-C++ file from this list ClassWizard will ignore it when rebuilding the .clw file.

Use this drop-down list to navigate to a different drive location if necessary when adding files.

Use this box to specify additional files that you want included in the .clw file for this project. By default, you can click OK to rebuild the .clw file using all files in the current project, without adding files.

Displays the files in the current directory location according to the List Files of Type filter. Initially displays all source files in the current project. To add files from a different project or location, use the Directories: list box to navigate to that location.

Use this list box to navigate to a different project or folder if necessary when adding files.

Lists the name of the class whose message map information is displayed in the dialog. To display another class, choose from the list box. Depending on the type of class displayed, the Object IDs list will display additional information about the class.

Lists the IDs of the selected object that can generate messages, such as menu items, dialog box controls, and so on. The first entry in this list is always the name of the current class.

Lists the messages that the selected object in the Object IDs list can handle and/or MFC virtual functions that may be overridden. To display Windows messages the window can receive, select the name of a class associated with a window. Messages in bold already have message-handler functions. Messages preceded by an equal sign (=) are the reflected messages of control classes. Reflected messages allow objects of control classes to handle their own messages.

Lists the member functions for the selected class. Member functions are message-handler functions or Microsoft Foundation Class (MFC) virtual functions. Items marked “V” are virtual functions, and items marked “W” are Windows messages.

Click to add a member function to the Member Functions list. Insert code for the actions that a member function takes when its corresponding object receives a message. This code defines the message handler for that object.

Click to delete the member function declaration from the header file and the function reference from the message map for the selected function.

Click to open an editor window . The insertion point will be at the selected member function.

Click to open a reference topic which describes the message in more detail. Available when a Windows message is selected in the Messages box.

Displays the purpose of the message handler or virtual function. Available when you have made a selection in the Messages box.

Lists the name of the class whose Automation properties are displayed for editing. Select a class that supports Automation and click Add Method and Add Property to add functionality. This list shows all classes and is not filtered for only classes that support Automation.

Lists the names of the methods and properties that you have added to the class. These are the names that objects of this class expose to Automation clients.

When enabled, makes the selected property the default property for the selected ActiveX object.

Click to add new Automation methods to your class. ClassWizard automatically updates the dispatch map of your class when you add or delete methods. Available only for classes that support Automation.

Click to add new Automation properties to your class. ClassWizard automatically updates the dispatch map of your class when you add or delete properties. Available only for classes that support Automation.

Click to delete the name of the method or property that is currently selected in the External Name list. If this property was implemented with the Get/Set type of implementation, you must manually delete the Get and Set member functions from your implementation file.

Click to open an editor window. The insertion point will be at the selected property or method.

Click to specify the level of data binding the Automation control supports.

Displays how the method or property that is selected in the External Name box is implemented in your C++ class: S = stock property, C = custom property, M = method, and bold typeface = the default property.

Lists the name of the class whose properties are displayed for editing. This list shows all classes and is not filtered for only classes that implement ActiveX controls.

Lists the names of the elements that you have added to the class. These are the names that objects of this class expose to Automation clients.

Click to add an Automation event handler. Events are used to communicate between the class and the container for the class.

Click to delete the selected Automation event handler.

Specifies the data source options.

When enabled, specifies that the data source is an ODBC (Open Database Connectivity) database.

Lists ODBC data source names. The wizard binds all columns of the table to the member variables of a **CRecordset**-derived class. Select a name in the list to choose a table for the data source.

When enabled, specifies that the application use a data link to an OLE DB data source.

Clicking this button invokes a wizard to help you create a connection to an OLE DB data source. You will need to specify the OLE DB provider, the location of the data, the data source, logon ID, and password.

When enabled, specifies that the data source is a DAO (Data Access Objects) database.

Specifies the DAO database name. Enter or select an existing database. By default, only Microsoft Jet (.MDB) databases are listed. To work with another database management system (DBMS), such as an ODBC data source, it is more efficient to attach tables from that source to an .MDB database than to open the database directly. For more information, see the article DAO External: Working With External Data Sources in *Programming With MFC*.

Specifies the Recordset options.

When enabled, specifies that the recordset will be a snapshot. A snapshot is the result of a query and is a view into a database at one point in time. All records found as a result of the query are cached, so you will not see any changes to the original records.

When enabled, specifies that the recordset will be a dynaset. A dynaset is the result of a query that provides an indexed view into the queried database's data. A dynaset caches only an integral index to the original data and thus offers a performance gain over a snapshot. The index points directly to each record that was found as a result of a query, and indicates if a record is removed. You will also have access to updated information in the queried records.

When enabled, specifies that the recordset will be created in a table. A table provides you with a means of directly manipulating the records and data in a database.

When enabled, sets **m_bCheckCacheForDirtyFields** to TRUE which creates a data cache to detect whether data values or NULL status has changed. For more information, see DAO RecordFieldExchange: Double Buffering Records in *Programming with MFC: Encyclopedia*.

When enabled, binds all unbound recordset field data members to the corresponding columns in a table on the data source. By default, AppWizard and ClassWizard bind all columns, so you seldom need to use Data Binding. If you have unbound some or all columns (by deleting their associated data members), you can rebind them all with Data Binding. Data Binding is only available for **CRecordset** derived classes.

Displays the column options for the data source.

Displays tables, views, queries, and stored procedures in the selected data source. Select the items to be included in the recordset.

Displays the tables contained in the selected data source. Select the tables for which you want recordsets created.

Lists the name of the class that you want to repair.

Lists the name of the Header (.H) file that contains the header information for the repaired class.

Lists the name of the file that contains the implementation code for the repaired class.

Click to remove the class from the ClassWizard list of classes.

Click to browse for the directory that has the header or implementation files that contain the class source code.

Lists the name of the **CRecordset** derived class you wish to associate with a **CRecordView** derived class. Select a recordset class from the list.

Click to create a new **CRecordset** derived class, where the class you select from the Class list is the base class for the new class.

Displays classes you can choose from to associate with the new resource. This makes the resource command IDs available for mapping when the class is selected in the ClassWizard Message tab or in WizardBar.

Displays the new dialog box, menu, toolbar, or accelerator resource which you can associate with an existing class. This makes the resource command IDs available for mapping when the class is selected in the ClassWizard Message tab or in WizardBar.

Click to associate the selected class with the new resource.

Displays the name you supplied for the add-in. This name also appears on the Add-ins and Macro Files tab of the Customize dialog box.

Displays the description you supplied of the add-in. This description also appears on the Add-ins and Macro Files tab of the Customize dialog box.

When selected, creates a toolbar button the user can click to carry out a command added by the add-in.

When selected, allows the add-in to respond to Visual C++ Developer Studio events.

Select this option to create a CHttpFilter-derived class. This creates a filter in which you can monitor and modify transaction data as it passes to and from the server and any of its clients.

The name of the filter class.

Specify a description that the wizard-generated code will register with the server when the filter initializes.

Select this option to create a CHttpServer-derived class. This class creates a server extension DLL that the Web server will call to execute code in response to requests from users of your Web site.

The name of the server extension class.

Specify a description that the wizard-generated code will register with the server when the server extension initializes.

Select this option to dynamically link to the MFC DLLs.

Select this option to statically link, importing the necessary routines from MFC.

Select high priority to cause your filter to be called first. This can significantly slow server performance.

Select medium priority to cause your filter to be called after all high-priority filters and before low-priority filters.

Select low priority to cause your filter to be called after all high- and medium-priority filters. Low priority is recommended because it has the least impact on server performance.

Specifies that your filter will be called for sessions on secure connections.

Specifies that your filter will be called for sessions on unsecured connections.

Specifies that your filter will be called after raw data is sent from the client and before the server processes it.

Specifies that your filter will be called after raw data is sent from the server and before the client receives it.

Specifies that your filter will be called when the server has preprocessed the client header.

Specifies that your filter will be called when the server is authenticating the client.

Specifies that your filter will be called when a logical URL is mapped to a physical path on your server. You can redirect the URL to another location.

Specifies that your filter will be called before a log write occurs. You can view, add, or change log information before it is written.

Specifies that your filter will be called when the session with the client is ending.

Application Frame

The Visual C++ application frame is displayed when all files are closed.

The status bar at the bottom of the frame displays the current status of Visual C++, the current status of an ongoing process, or the current action that you can take.

If you can edit in the currently-active window in Visual C++, the status bar indicates the following: the line and column position of the insertion point, if you are in column select mode, if you are in insert or overwrite mode (toggled with the INS key), if the file in the currently-active window is read-only, and if you are currently recording a macro. In addition, you can display a digital clock on the status bar.

Accelerator Editor Window

Use the Accelerator Editor window to add, delete, change, or browse the accelerator-key assignments in your project. You can right-click on this window to bring up a shortcut menu of commands.

Binary Editor Window

Use the Binary Editor window to edit an existing custom resource at the binary level in either hexadecimal or ASCII format. You can right-click on this window to bring up a shortcut menu of commands.

Image Editor Window

Use the Image Editor window within the Graphics editor to edit bitmaps, GIFs, JPEGs, icons, and cursors.

Note The Image Editor window does not have a shortcut menu because the right mouse button is used for image editing functions.

InfoView

The InfoView pane displays the table of contents for the online documentation. Click a plus or minus sign to expand or contract a book (or node). Double-click a topic to display it. You can right-click on this window to bring up a shortcut menu of commands.

Browse Window

Browse windows display information about the symbols in your program. When you build a project, you can specify that the compiler create a file with the information about the symbols in your project. This file has the project's base name and the extension .bsc. You can also browse through any .bsc file created with the bscmake.exe utility located in the directory with other executables.

You view browse information in browse windows, which have different appearances and different controls depending on the type of information they are displaying.

Symbol Type Abbreviations in the Browse Window
When you display symbols in the browse window, they are preceded by abbreviations denoting the type of symbol. The browse window uses the following abbreviations.

Abbreviation	Meaning
c	Class
f	Function
d	Data
m	Macro
t	Non-class type
V	Virtual function or data member
S	Static function or data member

Browse Window: Base Classes and Members

The Base Classes and Members view of the browse window displays the following information.

Window element	Function	Action
Left pane	Displays a graph of derivations.	Click the plus (+) or minus (–) sign to expand or contract the graph. Click the node or title to select it and display the corresponding information in the right panes. Double-click a symbol to open the source at the definition.
Top right pane	Displays member functions and member variables of the class selected in the left pane.	Click to display the definition and references in the bottom right pane. Double-click to open the source at the definition.
Bottom right pane	Displays definitions and references for the item currently selected in either the left or top right panes.	Double-click to open the source at the definition or a specific reference.
Push pin	Determines whether or not the window disappears after it loses focus.	Click to push or pull the pin.
Help button	Displays Help for the window.	Click for Help.
Filter lists	Filters the browser query to display selected types of information.	Select a filter type from the lists.

Browse Window: Call Graph

The Call Graph view of the browse window displays the following information.

Window element	Function	Action
Left pane	Displays a graph of functions that the selected function calls.	Click the plus (+) or minus (–) sign to expand or contract the graph. Click the node or title to select it and display the corresponding information in the right pane. Double-click to open the source at the definition or a specific reference.
Right pane	Displays definitions and references for the item currently selected in the left pane.	Double-click to open the source at the definition or a specific reference.
Push pin	Determines whether the window disappears after it loses focus.	Click to push or pull the pin.
Help button	Displays Help for the window.	Click for Help.

Browse Window: Callers Graph

The Callers Graph view of the browse window displays the following information.

Window element	Function	Action
Left pane	Displays a graph of functions that call the selected function.	Click the plus (+) or minus (–) sign to expand or contract the graph. Click the node or title to select it and display the corresponding information in the right pane. Double-click to open the source at the definition or a specific reference.
Right pane	Displays definitions and references for the item currently selected in the left pane.	Double-click to open the source at the definition or a specific reference.
Push pin	Determines whether the window disappears after it loses focus.	Click to push or pull the pin.
Help button	Displays Help for the window.	Click for Help.

Browse Window: Definitions and References

The Definitions and References view of the browse window displays the following information.

Window element	Function	Actions
Left pane	Displays the selected symbol or a list of the matching symbols if you used a wildcard.	Click to select a symbol from the list. Double-click to open the source at the definition.
Right pane	Displays definitions and references for the item currently selected in the left pane.	Double-click the location specified for the definition or reference to open the source at the definition or reference.
Push pin	Determines whether the window disappears after it loses focus.	Click to push or pull the pin.
Help button	Displays Help for the window.	Click for Help.

Browse Window: Derived Classes and Members

The Derived Classes and Members view of the browse window displays the following information.

Window element	Function	Action
Left pane	Displays a graph of derivations.	Click the plus (+) or minus (–) sign to expand or contract the graph. Click the node or title to select it and display the corresponding information in the right pane. Double-click a symbol to open the source at the definition.
Top right pane	Displays member functions and member variables of the class selected in the left pane.	Double-click to open the source at the definition.
Bottom right pane	Displays definitions and references for the item currently selected in either the left or top right panes.	Double-click to open the source at the definition or a specific reference.
Push pin	Determines whether the window disappears after it loses focus.	Click to push or pull the pin.
Help button	Displays Help for the window.	Click for Help.
Filter lists	Filters the browser query to display selected types of information.	Select filter types from the lists.

Browse Window: File Outline

The File Outline view of the browse window displays the following information.

Window element	Function	Actions
Left pane	Displays the functions and classes in the file by default.	Click to select a symbol from the list. Double-click to open the source at the definition.
Right pane	Displays definitions and references for the item currently selected in the left pane.	Double-click the definition or reference you want to see.
Push pin	Determines whether the window disappears after it loses focus.	Click to push or pull the pin.
Help button	Displays Help for the window.	Click for Help.
Filter buttons	Filters the browser query to display selected types of information.	Click to toggle the filter.

The left pane displays the following information about symbols.

Left entries	Middle entry	Right entry
Applicable filter types from the selected filters and virtual or static member indicator.	Symbol name	Type of symbol if the symbol name is ambiguous about type.

Call Stack Window

The Call Stack window lists the function calls that led to the current statement, with the current function on the top of the stack. By default, each call is shown with the parameter types and values passed to it. You can turn the display of parameter types and values off using the shortcut menu commands shown below.

If you double-click a frame (function) in the call stack, the debugger updates the windows as if you were in that frame. If you select a frame in the call stack and press F7, the program executes until it reaches that frame.

You can copy information from the Call Stack window and drag information to another window using the drag-and-drop feature.

You can right-click on this window to bring up a shortcut menu of commands.

Cursor Editor Window

You can edit icon and cursor bitmaps in the Graphics editor of Microsoft Visual C++.

Icons and cursors are like bitmaps, and you edit them in the same ways. But icons and cursors have attributes that distinguish them from bitmaps. For example, each icon or cursor resource can contain multiple images for different display devices. In addition, a cursor has a ‘‘hot spot’’ — the location Windows uses to track its position.

Note The Cursor Editor window does not have a shortcut menu because the right mouse button is used for image editing functions.

Dialog Editor Window

Use the Dialog Editor window to create dialog boxes, place and arrange controls, and test the finished dialog box. In addition, you can also add ActiveX controls, import Visual Basic forms to a dialog box resource, and save dialog boxes as templates for future dialog boxes. For easier layout, guides or a grid helps align groups of controls. You can right-click on this window to bring up a shortcut menu of commands.

Disassembly Window

Use the Disassembly window to view the assembly-language instructions that the compiler generates for your source code. You can right-click on this window to bring up a shortcut menu of commands.

Icon Editor Window

You can edit icon and cursor bitmaps in the Graphics editor of Microsoft Visual C++.

Icons and cursors are like bitmaps, and you edit them in the same ways. But icons and cursors have attributes that distinguish them from bitmaps. For example, each icon or cursor resource can contain multiple images for different display devices. In addition, a cursor has a ‘‘hot spot’’ — the location Windows uses to track its position.

Note The Icon Editor window does not have a shortcut menu because the right mouse button is used for image editing functions.

Memory Window

The Memory Window displays memory contents starting at a specified address (0x00000000 by default). A toolbar at the top of the window displays the starting address for the memory display. Edit the value in the toolbar and press RETURN to change the starting address. Use the scrollbar at the side of the window to view other memory locations in the program's address space without changing the starting address of the display.

If you drag a memory address from another window to the Memory window, the Memory window displays the memory contents starting at that address. If you drag a pointer from another window to the Memory window, the Memory window displays the memory contents starting at the address the pointer points to.

You can right-click on this window to bring up a shortcut menu of commands.

If you selected hexadecimal display in the Debug tab of the Options dialog box (Tools menu), the memory contents and the address in the **Address** box appear in hexadecimal format. If you edit the address in the **Address** box, you must enter the address in hexadecimal or use the prefix 0n (zero-n) to indicate that the number is a decimal (for example: 0n1000).

Menu Editor Window

Use the Menu Editor window to edit your program's menu resources. You can right-click on this window to bring up a shortcut menu of commands.

Output Window

The Output window displays several types of information. The Output window has a separate tab output from each tool — such as **Build**, **Debug**, and **Find in Files** (1 and 2). To display the output from a given tool, select the tab at the base of the Output window. You can copy and print information from the Output window.

Note To enable autoscrolling in the Output window, place the cursor on the last line.

If you click the **Build** tab, the Output window displays progress and error messages from the compiler and linker. The list includes all errors that prevent a program from building, with filename, line number, and error number. To find the source code corresponding to an error, select the error, then click the right mouse button and click **Go To Error/Tag** on the shortcut menu. If you display the status bar, it gives a summary of the current error.

In the Output window, you can display the results of two Find In Files searches. You can display the results of one search on the **Find in Files 1** tab, and you can display the results of another search on the **Find in Files 2** tab. In the **Find in Files** dialog box (**Edit** menu, **Find in Files** command), you specify which tab you want the results to go to. If you select the **Output to pane 2** check box, the results go to the **Find in Files 2** tab; otherwise, they go to the **Find in Files 1** tab.

If you click the **Debug** tab, the Output window displays messages generated by the debugger.

You can right-click on this window to bring up a shortcut menu of commands.

FileView

The FileView pane shows relationships among the projects and files included in the project workspace. The relationships in FileView are logical relationships, not physical relationships, and do not reflect the organization of files on your hard disk.

The icons used also give you information about the files.

If you have installed a source-code control system that conforms to the Microsoft Common Source Code Control Interface, the icons also represent some source-code control states. Grayed means that a file is under source-code control. A check next to the icon for a file under source-code control indicates that you have the file checked out.

In projects for programming languages, FileView shows the relationships of the source files and the dependent files used to build all project configurations. The active project in the workspace is indicated in FileView with bold type. The active project is the project that will be built when you use the commands **Build** or **Rebuild All**. You can select a different active project by using the **Set Active Project** command on the **Project** menu. The active configuration determines which set of build options is used when you build the active project. You can select a different active configuration by using the **Set Active Configuration** command on the **Build** menu.

Using FileView with the Visual Database Tools

Important: Feature Only Available with Visual Database Tools Database projects and related features are supported only when the Visual Database Tools are installed. These features include the Data View pane and the use of FileView to view items in database projects. Visual Database Tools are included in Visual C++ Enterprise Edition.

The FileView pane provides a graphical environment for creating, viewing, and editing local files in a database project. Your database projects appear in the FileView pane as a tree.

In the FileView pane you can:
- View the files and folders in your database projects.
- Manage your files. For example, you can add, move, rename, copy, and delete files.

The FileView pane lists the following items:
- Database projects
- Data connections
- Database queries

The FileView pane also lists all of the files and folders for each data source in your database projects. You can expand each folder to see the files that are stored inside. This feature is especially useful for managing your files. For example, you can:
- Open a folder that contains a file you want to move or copy.
- Select a file and move or copy it into another folder.
- Double-click a file to open it in the application workspace.
- Create, rename, or delete files and folders.

If you want to view only files and folders for a specific database project, you can collapse the database projects that you don't want to see. You can also choose which project you want to set as the active database project.

In addition to the FileView pane, you can use the Data View pane to create, view, and edit files residing on a remote database server as part of a database project. The Data View pane is visible as a tab in the Project Workspace window when your database project is connected to a data source. The Data View pane is available only with the Visual Database Tools.

Registers Window

The Registers window displays the names and current values of the native CPU registers and flags. It also displays the floating-point stack. You can change the value of any register or flag in the Registers window while the program is being debugged. You can right-click on this window to bring up a shortcut menu of commands.

Resource Browser (RC) Window

The Resource Browser (RC) window displays resources in a resource script file. Expanding the top-level folder shows the resource types. Expanding each resource type shows the individual resources of that type. You can right-click on this window to bring up a shortcut menu of commands.

String Table Editor Window

You can use the String Table Editor window to edit a string table resource. A string table resource contains a list of IDs, values, and captions for all the strings in your application. An application can have only one string table. You can right-click on this window to bring up a shortcut menu of commands.

Text Editor Window

The Text Editor window displays text files of any type, such as language source and header files.

You can right-click on this window to bring up a shortcut menu of commands. **Note** The shortcut menu commands change depending on the type of file you are editing.

Text Tool Window

Use the Text Tool window to add text information to a cursor, bitmap, or icon resource.
The text area of the window contains the text that appears as part of the resource. Initially this area is empty.

Click the **Font** button to change the font, style, or size of the cursor font.

The Text Tool window uses the default shortcut menu, which contains a list of windows and toolbars and the **Toolbars** and **Customize** commands.

Variables Window

The Variables window contains three tabs:
- The **Auto** tab displays information about variables used in the current statement and the previous statement.
- The **Locals** tab displays information about variables that are local to the current function.
- The **this** tab displays information about the object pointed to by **this**.

Each tab contains a spreadsheet field with three resizable columns. The debugger automatically fills these columns with the type, name, and value of variables appropriate to the tab.

A toolbar located above the tabs contains a drop-down list for specifying the current scope of the variable display. This toolbar can be hidden, or redisplayed, using the shortcut menu.

The Variables window replaces the Locals from previous versions of Visual C++ and adds new functionality.

If the Variables window contains an array, object, or structure variable, a button appears next to the variable name. By clicking the button, you can expand or contract your view of the variable. The button displays a plus sign (+) when the variable is displayed in contracted form, and a minus sign when it is displayed in expanded form.

The Variables window supports editing. You can cut, copy, or drag information from the Variables window. You can edit the Value column to change the value of a variable while debugging. If you selected hexadecimal display in the Debug tab of the Options dialog box (Tools menu), you must enter the value in hexadecimal or use the prefix 0n (zero-n) to indicate that the number is a decimal (for example: 0n1000).

Row and Column Behavior
To autosize a column to fit its contents, double-click on the vertical divider at the column edge. To size a column manually, drag the right divider to the left or right.

Note Rows are sized to fit the current font and cannot be resized manually. To change the font size, use the **Fonts and Colors** tab of the **Options** dialog box (**Tools** menu).

Auto Tab
The **Auto** tab displays information about variables from the current statement and the previous statement. Variables appear in alphabetical order. If a statement spans multiple lines, the **Auto** tab displays variables from the lines corresponding to that statement, up to a 10-line limit.

Locals Tab
The **Locals** tab displays the names, values, and types of all local variables in the current function. As you trace through a program, new variables come into scope.

this Tab
The **this** tab displays type, name, and value information about the object pointed to by the pointer **this**. All base classes of the object are automatically expanded.

You can right-click on this window to bring up a shortcut menu of commands.

Version Information Editor Window

Use the Version Information Editor window to create and maintain a version information resource for an application. Version information consists of company and product identification, a product release number, and copyright and trademark notification. You can right-click on this window to bring up a shortcut menu of commands.

Watch Window

The Watch window contains four tabs: **Watch1**, **Watch2**, **Watch3**, and **Watch4**. Each tab contains a spreadsheet field that displays variable information. You can enter variable names into the Name column of this field; the debugger fills in the Type and Value columns with the corresponding information as the program runs.

Note You can enter expressions, as well as variable names, in the Watch window's Names columns. The debugger evaluates the expression continuously as your program executes.

If the Watch window contains an array, object, or structure variable, a button appears next to the variable name. By clicking the button, you can expand or contract your view of the variable. The button displays a plus sign (+) when the variable is displayed in contracted form, and a minus sign when it is displayed in expanded form.

The Watch window supports editing functions. You can cut or copy information from the Watch window. You can cut or copy a variable from another window and paste it into the Watch window or drag it in using a drag-and-drop operation. You can edit the Value column to change the value of a variable while debugging. If you selected hexadecimal display in the Debug tab of the Options dialog box (Tools menu), you must enter the value in hexadecimal or use the prefix 0n (zero-n) to indicate that the number is a decimal (for example: 0n1000).

Row and Column Behavior
To autosize a column to fit its contents, double-click on the vertical divider at the column edge. To manually size a column, drag the vertical divider at the edge of the column.

Note Rows are sized to fit the current font. To change the font size, use the **Fonts and Colors** tab of the **Options** dialog box (**Tools** menu).

You can right-click on this window to bring up a shortcut menu of commands.

InfoViewer Topic Window

The InfoViewer Topic window displays a topic from the InfoViewer Help system.

This window offers a toolbar for moving within the Help system; with this toolbar, you can jump to the previous or next topic in the Help system, or synchronize the Table of Contents (in the Help pane of the Project Workspace window) to show you where the current topic resides within the Help system.

ClassView

The **ClassView** tab of the Project Workspace window displays the classes and their members for all projects on the workspace. The folder name shown in bold type represents the default project configuration. Expanding a project folder displays the classes included in that project. Expanding a class displays the members in that class. Double-clicking a member takes you to its location in source code. ClassView also displays COM interfaces that have been defined in an IDL or ODL file for the project.

The **ClassView** shortcut menu changes dynamically depending on whether the focus is on a project, class, or class member. When focus is on a project, you can use the **ClassView** shortcut menu to add a new class. When focus is on a class, you can add members to the class. When focus is on a COM interface, you can add properties and methods to the interface. For ActiveX control interfaces, you can also define new events.

Default Shortcut Menu

The shortcut menu shows commands you can use to customize your development environment. This menu contains the following commands.

Shortcut menu command	Description
Output	Shows/hides the Output window.
Watch (if debugging)	Shows/hides the Watch window.
Variables (if debugging)	Shows/hides the Variables window.
Registers (if debugging)	Shows/hides the Registers window.
Memory (if debugging)	Shows/hides the Memory window.
Call Stack (if debugging)	Shows/hides the Call Stack window.
Workspace	Shows/hides the Workspace window.
Standard	Shows/hides the Standard toolbar.
Build	Shows/hides the Build toolbar.
Build MiniBar	Shows/hides the Build MiniBar toolbar
Resource	Shows/hides the Resource toolbar.
InfoViewer	Shows/hides the InfoViewer toolbar.
Edit	Shows/hides the Edit toolbar.
Debug	Shows/hides the Debug toolbar.
Browse	Shows/hides the Browse toolbar.
Database	Shows/hides the Database toolbar.
Wizard Bar	Shows/hides the Wizard Bar toolbar.
Customize	Displays the **Customize** dialog box, in which you can customize your development environment to a greater extent.

Toolbar Editor Window

Use the Toolbar Editor window to edit toolbar resources. This Toolbar editor also converts bitmaps into toolbar resources. There is also a menu command on the **Image** menu to switch between the Graphics editor and the Toolbar editor.

Note The Toolbar Editor window does not have a shortcut menu because the right-mouse button is used for image editing functions.

ResourceView

ResourceView displays the resource files included in your projects. Expanding the folder shows the resource types. Expanding each resource type shows the individual resources of that type. You can right-click on this window to bring up a shortcut menu of commands.

Binary File Properties: General

Displayed only when the Binary editor window is active.

Filename
Displays the full pathname of the binary file currently viewed in the Binary editor.

Displays a brief description for the selected component.

Click this button to display complete help for the selected component.

Displays the path to the file containing the selected ActiveX control.

Specifies the number of ActiveX controls that your project will create. The maximum is 99.

Specifies that your control will be licensed. The ActiveX ControlWizard inserts several function calls and generates a .lic file.

Select this option if you do not want to include licensing support for your controls.

Select this option if you want the ActiveX ControlWizard to insert comments in the source and header files that indicate where you need to add your own code. This option is enabled by default.

Select this option if you do not want comments included in the files generated.

Select this option if you want ControlWizard to generate a set of help files that provide context-sensitive help. Help support requires using the Help compiler, which is provided with Visual C++.

Select this option if you do not want to generate help files.

Select one of the controls in your project. Default names of the controls are created from the project name you specify in the Name box on the New Project Workspace dialog box.

Click to modify the default names associated with the currently-selected control.

Select this box to have the currently-selected control indicate to its container that the control prefers to be automatically activated when it is visible. The container is not required to support this request.

Select this box to have the currently-selected control indicate to its container that the control should be invisible in the container's run-time mode and visible in the container's design-time mode. A container may ignore the control's preference in which case the control will be visible at all times.

Select this box to have the currently-selected control listed in the Insert Object dialog box of a container application.

Select this box to create a standard About dialog box and AboutBox method for the currently-selected control. The About dialog box is displayed when your control's AboutBox method is invoked by the container.

Select this box to have the currently-selected control support ISimpleFrameSite. When the container and the control both support this protocol, the container uses simple-frame controls as parents for other controls in the container. In effect, the simple frame control operates as a compound document container, but the frame control's container does nearly all the work.

Choose a common Windows control from the drop-down list, such as a button, toolbar, or edit box, to subclass.

Click the Advanced button to display the Advanced ActiveX Features dialog box. In the dialog box, you can choose windowless, flicker-free, asynchronous, and other ActiveX control optimizations.

The base name of the control. The default is based on the name of the project. If you change the default, the ControlWizard changes the other names that appear on this dialog box.

The name of the C++ class that represents the control.

The name of the header file (.h) that contains the control's class definition.

The name exposed to the programmer from an Insert Object list.

The name of the source file (.cpp) that contains the class implementation.

The ID of the control class. This is the string that a control registers in the registry when it is added to a project. This string is used by a container application to create an instance of a control.

The C++ class that manages the user interface for viewing and editing the properties of a control.

The name of the header file (.h) that contains the class definition.

The name exposed to the programmer. The property page user type name is rarely used.

The name of the source file (.cpp) that contains the class implementation.

The ID of the property page class. This is the string that a control registers in the registry when it is applied to a project. This string is used by a container application to create an instance of a control's property page.

Select this option to use windowless activation. Use when a control does not need a window of its own and can use the window services of its container. Required for transparent or nonrectangular controls.

Select this option to disable tests for clipping (which will improve speed). Select only if you are certain the control doesn't paint outside its client rectangle. Not available for windowless controls.

Select this option to eliminate drawing operations and accompanying visual flicker in transition between inactive and active states. Select only if the control draws itself identically in the inactive and active states. Not available for windowless controls.

Select this option to allow your control to process WM_SETCURSOR and WM_MOUSEMOVE messages when it is not active. The container delegates messages to IPointerInactive, which dispatches the messages through your control's message map.

Select this option to indicate that the control wishes to perform optimized drawing if the container supports it.

Select this option to specify that your control can have properties that point to data that is loaded in the background.

Specifies that your server is a Dynamic Link Library (DLL) and therefore an in-process server.

Specifies that your server is an Executable (EXE) and therefore a local out-of-process server.

Specifies that your object includes MFC support, and the MFC libraries will be attached to your code (available for .DLL projects only).

Specifies that proxy/stub code be merged into your main project (available for .DLL projects only). By default, the proxy/stub marshaling code is contained in a separate DLL.

Specifies that the server is also a Windows NT service that runs in the background when NT starts up.

Specifies the support for Microsoft Transaction Server by adjusting the project build settings.

This setting specifies the target platform for which your project is compiled. This setting changes the value of the /QA compiler switch. The default setting uses a blended model that runs equally well across all Alpha CPUs. You must manually modify this compiler option to optimize your application for a specific Alpha processor.

This option determines whether the program being built has single or multi-thread support. This setting changes the value of the /MD, /ML, and /MT compiler switches. It is usually best to use the multithreaded options when building a DLL. If you specify a single-threaded option, your DLL will work reliably only when called by single-thread applications.

A calling convention defines the way internal and external functions are called. Calling conventions differ in how arguments are passed, how values are returned by functions, and how the stack is cleaned up. __cdecl is the default calling convention for C and C++ programs and is the only convention supported on the Alpha platform.

This setting allows you to use non-standard alignments for data structure members. Alignments other than the default can cause severe application performance degradation due to alignment faults and subsequent fixups. Change this setting from the default with caution. This setting changes the value of the /Zp compiler switch.

This setting defines the maximum size of a data element that can be placed in the Global Pointer register for fast subsequent access. Data in the gp area can be accessed with one machine instruction rather than two. This setting changes the value of the /GT compiler switch. For best results, set this to 32 when producing RISC applications, and to zero (the default) when building DLLs.

This setting is ignored by the Alpha version of Visual C++.

This setting specifies the target platform for which your project is compiled. This setting changes the value of the /Q compiler switch.

A calling convention defines the way internal and external functions are called. Calling conventions differ in how arguments are passed, how values are returned by functions, and how the stack is cleaned up. Refer to the online reference material for the calling conventions applicable to the x86 platform.

This setting allows you to use non-standard alignments for data structure members. Alignments other than the default can cause severe application performance degradation due to alignment faults and subsequent fixups. Change this setting from the default with caution. This setting changes the value of the /Zp compiler switch.

This option determines whether the program being built has single or multithread support. This setting changes the value of the /MD, /ML, and /MT compiler switches. It is usually best to use the multi-threaded options when building a DLL. If you specify a single-threaded option, your DLL will work reliably only when called by single-thread applications.

This option creates a DLL project with the specified name but adds no files to it. Use this when you intend to supply all your own source files.

This option creates a .dsp file based on the project name you chose, containing a .cpp file with a blank DllMain function. It also adds stdafx.cpp and stdafx.h files to the project and sets compiler options to create the precompiled header file.

This option creates a DLL-equivalent of a simple “Hello World” application, which demonstrates exporting symbols (a function, variable, and class) from a DLL.

This option creates a project (.dsp) file with the specified name but adds no files to it. Use this when you intend to supply all your own source files.

This option creates a .dsp file based on the project name you chose. The application files include a.cpp file with a empty main function, and a precompiled header file (stdafx.h).

This option creates a simple console-based ‘‘Hello World’’ application with supporting code in .cpp and .h files. The application prints ‘‘Hello, World!’’ to the console and then exits.

This option creates a console application that contains code to initialize the Microsoft Foundation Class Libraries (MFC). These libraries include support for non-Windows applications, such as **CString** for string implementations, MAPI for mail support, Collection classes to create lists, and so forth.

This option adds stdafx.cpp and stdafx.h files to the project and sets compiler options to create a precompiled header using stdafx.h.

This option sets the compiler switches required for MFC and, if you've checked the pre-compiled header file option, also adds MFC header files to the precompiled header file.

Same as IDD_SELECT_TARGET.

Identifies files for which you want to specify custom tools.

Specifies the build step and is displayed on the Build tab of the Output window during the build.

Enter the commands to run. If you enter more than one command, the build system runs them in order from top to bottom. You can use the directory and file macros in these commands.

Enter the names of the output files that you create with the commands in the Commands list. The build system checks these files during a build to determine whether they are out of date with respect to the input file. If so, the build system builds them.

Lists directory macros that you can insert at the current insertion point in the Commands or Outputs list. Selecting from the list inserts the macro into the command or name that you are currently entering.

Lists filename macros that you can insert at the current insertion point in the Commands or Outputs list. Selecting from the list inserts the macro into the command or name that you are currently entering.

Click to identify dependency files for this project that are generated by custom build rules.

Provides a space for you to enter or select the project with which you want to work.

The path and name of the executable file that you want to debug. This location is always relative to the project's directory, as defined by where the .dsp file for each project exists on your local hard drive.

Click to display an Open dialog box that you can use to locate the executable you want to debug.

Type the name to identify the new configuration.

Choose the existing configuration from which to copy initial settings.

Choose the platform to be used in the new configuration. This choice may alter some settings from the settings initially copied and may specify a different tool set.

Enter the new name for the folder here.

Filters files into a group when they are inserted into the project. For example, if you set the File extensions field to bmp;gif;jpg for the Resource Files group and then add files with those extensions to the project, the files will end up in the Resource Files group.

Note: You can add files without the file extensions defined for a group by using either the context menu or a drag-and-drop operation.

Displays the existing projects and configurations in the open project workspace.

Displays the Add Configuration dialog box with which you can add a configuration to the project selected in the Projects and Configurations tree. You can select a project by selecting either the project node or a configuration in the project.

Removes the selected configuration from the project. If you remove the last remaining configuration in a project, the project is also removed.

The path and name of the remote executable file that you want to debug. The debugger will interpret this path from the perspective of the remote machine.

Click to export makefiles for all listed projects.

Select this option to force the writing of makefile dependency files. These files contain the source file dependency information (for example, <MyFile.cpp> is dependant on <DepFile1.h> and <DepFile2.h>). Selecting this option significantly slows the export of a makefile and is generally not necessary when only compiler switches have changed.

Choose the configuration(s) with which to work.

Select a project from the list of existing projects. You add the subprojects that you select or create to this project, and/or remove subprojects from this project.

Select any projects on which the current project depends.

Choose a category from the drop-down list.

Shows the switches the tool will use to build. If a project node is selected in the settings tree, you can type into this field to add or remove switches.

Same as IDD_TOOL_GENERAL.

Same as IDD_SELECT_TARGET

Lists user-defined dependencies. You can add or delete dependencies, or reorder the list using the four buttons at the top right of the list.

If this check box is selected, the debugger asks for additional DLLs when debugging begins.

To make a row active, select any cell in that row by double-clicking. When a row is active, a button with an ellipsis (...) appears in the rightmost cell. Preloading is useful if you want to set breakpoints in a DLL that is not loaded implicitly using IMPLIB or at startup. You can use this option to preload symbols even after debugging has begun.

This option will always export a makefile when saving the current project.

Writes a text log of the build, named ACTIVEPROJECTNAME.plg, to the active project directory.

Specifies the platform on which the directory paths exist.

Determines the type of directories displayed in the Directories list box.

A list of directories to find project files, in hierarchical order. Add directories or change the order to load all the necessary support files for your project.

Displays the text that is echoed to the build Output window when this step is executed.

Displays the list of commands to execute. You can add new commands, delete commands, and reorder commands using the four buttons on the top right of the list.

Displays the text that is echoed to the build Output window when this step is executed.

Displays the list of commands to execute. You can add new commands, delete commands, and reorder commands using the four buttons on the top right of the list.

Specifies a name and/or location for the Object Description Language file (.odl) other than the default. This option is available only if you select an .odl file in the Source Files folder of the Settings For list box.

Generates an include file (.h) that contains the types definitions from the Object Description Language file (.odl). The generated file can be included in a C or a C++ file. This option is available only if you select an .odl file in the Source Files folder of the Settings For list box.

Specifies additional directories in which to search for include files.

Define one or more preprocessor macros for use with the #ifdef directive.

Prevents the display of the MkTypeLib startup banner.

When selected, includes the **/mktyplib203** switch on the MIDL compiler. This option is not recommended unless you specifically have a problem with a particular .odl file.

Select this option to generate the smallest proxy stub code while still delivering good performance. This is the optimal setting for projects targeted for the Windows NT 4.0 or DCOM95 platforms. Do not use this setting for NT 3.51 or default Windows 95 projects. Equivalent to the **/Oicf** compiler switch.

Select this option to generate a .c file that contains definitions for all the GUIDs (interface IDs) in the IDL file. Include this .c file, for example, in a .cpp file to obtain the GUIDs for the COM objects defined in this type library. Equivalent to the **/iid** compiler switch.

Same as IDD_TOOL_GENERAL.

Exclude the project dependency from the build.

Click this to restore all the settings to their original values.

When selected, allows you to use a custom build step for a particular type of file, instead of the default tool associated with that file, such as a c compiler for a .cpp file, or the Resource compiler for an .rc file. You can define the properties for the custom build step by clicking Settings on the Project menu, selecting the file, clicking the General tab, and clicking Always use custom build step. Then click the Custom Build tab.

This will exclude the selected file(s) from the build.

Specifies how your program is linked to MFC.

Specifies where files used to build your program, such as object and source files, are located. This location is always relative to the project's directory, as defined by where the .dsp file for each project exists on your local hard drive.

Specifies where final output files are located. This location is always relative to the project's directory, as defined by where the .dsp file for each project exists on your local hard drive.

Allows exported makefile dependencies to be written per project configuration, if necessary. Most projects will have the same dependencies for all configurations, unless they are multi-platform or have conditional #include directives. For better performance, leave this option clear unless it is required to generate the correct dependencies. If this option is not selected, the compiler uses the same dependency settings for all configurations.

Lists the current project or projects. Click the plus sign to display the folders and files contained in the project; click the minus sign to collapse the display. Depending on what you select, you can set options for individual files, folders, or the entire project.

Lists the configuration(s) for which you are creating or modifying settings. Click the arrow to display the available configurations.

Lists the configuration(s) for which you are creating or modifying settings. Click the arrow to display the available configurations.

The selected items have no properties in common.

Provides a place for you to type the command line that the operating system executes for this project when you click Build on the Build menu. By default, the system executes Microsoft NMAKE with the /F option, followed by the name of the external makefile in the project subdirectory that you have just created. You can revise this field to execute any command, such as any batch file. You could create a batch file, for example, that first changes to the directory with the external project files and then runs NMAKE.

Provides a place for you to type the options added to the command line when you click Rebuild All on the Build menu. By default, /A for Microsoft NMAKE is added.

Provides a place for you to type the name of the file that is created when you build a project.

Provides a place for you to type the name of the browse information file that is created when you build this project. It must have the file extension .bsc.

Lists the installed platforms to create external projects for those platforms.

Provides a place for you to type the name(s) of one or more preprocessor macros to create.

Provides a place for you to type the name of a macro to undefine.

Select this option to specify that every previously-defined macro will be undefined.

Provides a place for you to add one or more directories to the list of directories searched for include files. Separate directory names with a comma.

Select this option to prevent the compiler from searching for include files in directories specified in the PATH and INCLUDE environment variables.

Select this option to generate intermediary (.sbr) browse files when you build your program. To create the .bsc file required for browsing, select the Build Browse Info File check box on the Browse Info tab.

Provides a place for you to specify a directory and/or filename for the .sbr and .bsc files generated by selecting the Generate Browse Info option.

Select this option to generate browse information files containing complete symbolic information, minus information on local variables, that you can examine in browse windows.

Specifies the generation of a listing file.

Provides a place for you to specify a directory and/or filename for the listing file selected from the Listing File Type list box.

Specifies one of four predetermined optimization categories: Default, Disable, Maximize Speed, and Minimize Size. If you specify Customize from this, the General category, you can use a list box in the Optimizations category to specify a custom set of optimizations.

Select from a list of compiler code optimizations.

Lists the type of functions that you can suggest for the compiler to expand inline.

Select this option to disable precompiled headers.

Select this option to create and use precompiled header files using the Automatic Use of Precompiled Headers system.

Select this option to create a precompiled header file (.pch). Select the Use precompiled header file (.pch) option to use the created .pch file.

Select this option to use a precompiled header file (PCH).

Provides a space for you to specify the name of a header file (.h) included in the selected source file. If a precompiled header (PCH) exists and matches the source code up to and including this header, then the compiler uses that PCH; otherwise, the PCH is replaced with one that matches the source code up to and including this header.

Provides a space for you to specify the name of a header file (.h) included in the selected source file. The compiler creates a precompiled header (PCH) from the source code up to and including this header. Other source files in this project can use this precompiled header.

Provides a space for you to specify the name of a header file (.h) included in the selected source file. The compiler ignores the source code up to and including this header and uses the contents of the precompiled header (PCH) instead. If no PCH exists, then an error will occur.

Select this option if you want the compiler to use ANSI C rules. Clear this option if you want to use Microsoft C language extensions.

Select this option to exclude and/or order individual functions in a DLL or executable file. You cannot modify this option (it is enabled by default) if you have selected Edit and Continue in the Debug info drop-down list (under the General category).

Select this option (equivalent to setting the **/Gf** compiler switch) to place a single copy of identical strings into the executable file. This feature is also called string pooling. You cannot modify this option if you have selected Edit and Continue in the Debug info drop-down list (under the General category). In this case, **/GF** becomes the default compiler switch, and the pooled strings are placed in read-only memory.

Select this option to detect changes to C++ class definitions and to recompile the source files.

Select this option to enable function-level recompilation.

Select this option to suppress display of the sign-on banner and informational messages.

Specifies the method used by the compiler to represent pointers to class members.

Specifies the inheritance model of classes. When the representation method is General-Purpose Always, you must also specify a value here.

Select this option to call destructors for automatic objects during a stack unwind caused by either a Windows NT-based structured exception or a C++ exception.

Select this option to add code for checking object types at run time (run-time type information).

Select this option to suppress the vtordisp constructor/destructor displacement member if you are certain that all class constructors and destructors call virtual functions virtually.

Specifies the severity of warning for which the compiler generates messages.

VPROJ_HIDC_OPTIMIZE(text derived from other control … **

Select this option to emit warning messages as error messages.

Provides a place for you to type the name(s) of one or more preprocessor macros to create.

Specifies the type of debugging information generated by the compiler. You must also change linker settings appropriately, depending on the option you specify here. These options correspond to the following compiler switches: C7 Compatible = /Z7; Line Numbers Only = /Zd; Program Database = /Zi, Program Database for Edit and Continue = /ZI.

Provides a space for you to specify an output file and to override the default name and location of the program that the linker creates.

Provides a space for you to specify an object file or standard library (either static or import) to pass to the linker. Separate file names with a space.

Select this option to generate debugging information for the executable file or DLL.

Select this option to remove all default libraries from the list of libraries the linker searches when resolving external references.

Uses text from this CHID in LINKER_CUSTOM.

Select this option to create a mapfile.

Select this option to create an output file that can be used with the Microsoft 32-Bit Source Profiler.

Select this option for a DLL subproject that does not generate an export library. If you do not select this option and an export library is not generated, the parent project will not build. If you select this option and an export library is generated, the build system ignores the library.

Select this option to create a mapfile. (Same ID as Proj_Linker dialog.)

Specifies a name for a mapfile other than the default.

Select this option to create a mapfile.

Select this option to generate new-style Microsoft debugging information.

Select this option to generate COFF-style debugging information.

Select this option to generate both COFF debugging information and old-style Microsoft debugging information.

Select this option to distribute debugging information in the source (compiler) .pdb files. Linking can be faster with this option but debugging can be slower due to the need to locate more files. Do not use this build when more than one user will debug the build.

Specifies where final output files are located.

Select this option to suppress the display of the startup banner.

Provides a space for you to specify the name of the output (.res) file. Visual C++ assumes a path relative to the project directory.

Lists the natural language (such as US English, Australian English, or Mexican Spanish) used in the resource.

Provides a space for you to specify one or more directories to add to the list of directories searched for include files. Separate path names with a comma.

Select this option to prevent the compiler from searching for include files in directories specified in the PATH and INCLUDE environment variables.

Provides a space for you to specify one or more preprocessor macros to create.

Cleans and builds the specified configuration(s), including all dependent projects. Equivalent to the /REBUILD option in a command-line build. Compare to the Clean command, which deletes intermediate and output files but does not rebuild the project or configuration.

Deletes all intermediate and output files created during the build process for the specified configuration(s). Does not build the project. These files are located in the directories defined on the General tab of the Project Settings dialog box, under "Intermediate files" and "Output files". Equivalent to the /CLEAN option in a command-line build.

Builds the specified project configuration(s) without building any dependent projects. Equivalent to the /NORECURSE option in a command-line build.

VPROJ_HIDC_NEW_ID (This topic is a placeholder for the new ID requested for the Build control in this dialog.)** Builds the specified configuration(s). Equivalent to the /MAKE option in a command-line build.

Provides a space for you to specify an object file or standard library (either static or import) to pass to the linker. Separate file names with a space.

Provides a space for you to specify one or more default libraries to remove from the list of libraries the linker searches when resolving external references. Separate file names with a comma.

Select this option to remove all default libraries from the list of libraries the linker searches when resolving external references.

Provides a space for you to specify one or more symbols to add to the symbol table. Forces linking with the object that contains the symbol definition(s).

Provides a space for you to specify additional paths that the linker can use to search for libraries. Separate each path specification with either a comma or semicolon. These paths are in addition to the path specified in the Library files category located on the Directories tab of the Options dialog box (Tools menu).

Select this option to place debugging information in a program database (PDB).

Select from the available options for incremental linking. Not available unless ‘Use program database’ is selected. Enable incremental linking for faster linking and Edit and Continue in debug builds; disable for release builds.

Provides a space for you to specify the filename for the program database (PDB).

Provides a space for you to specify an output file and override the default name and location of the program that the linker creates.

Specifies that you want to generate a valid executable file or DLL even if a symbol is referenced but not defined or is multiply defined.

Select this option to display details about the linking process during a build.

Select this option to suppress the display of the copyright message and version number during a build.

Provides a space for you to specify the name of the browse information file that is created when you build this project. It must have the file extension .bsc.

Select this option to build a browse information file (BSC) each time you build a program. Make sure the Generate Browse Information check box on the appropriate language tab is selected also, so that for each source file, the compiler generates .sbr files used to create the browse information file. Click the See Also button for a list of topics that include information about browse information settings.

Select this option to suppress the display of the startup banner.

Provides a space for you to specify the directory in which debugging occurs. If you do not specify a directory, debugging occurs in the directory where the executable is located.

Provides a space for you to specify command-line arguments you want to pass to the program at startup. The program receives these arguments when started with the Go or Restart command. You can use I/O redirection in this field using cmd.exe format.

Provides a space for you to specify a base address for a program. Specified address overrides the default location for an executable file (at 0x400000) or a DLL (at 0x10000000).

Provides a space for you to specify a starting address for an executable file or DLL.

Provides a space for you to specify the total stack allocation, in bytes, of virtual memory. The default stack size is 1 MB.

Provides a space for you to specify the amount, in bytes, of physical memory to allocate from the reserve memory.

Provides a space for you to specify the major part of a version number, which precedes the decimal point, to insert into the header of an executable file or DLL.

Provides a space for you to specify the minor part of a version number, which follows the decimal point, to insert into the header of an executable file or DLL.

Provides a place for you to type the command line that the operating system executes for this project when you click Build on the Build menu. By default, the system executes Microsoft NMAKE with the /F option followed by the name of the external makefile in the project subdirectory that you have just created. You can revise this field to execute any command, such as any batch file. You could create a batch file, for example, that first changes to the directory with the external project files and then runs NMAKE.

Provides a place for you to type the options added to the command line when you click Rebuild All on the Build menu. By default, /A for Microsoft NMAKE is added.

Provides a place for you to type the name of the file that is created when you build a project.

Provides a place for you to specify the name of the browse information file that is created when you build this project. It must have the file extension .bsc.

Specifies the platform for an external makefile loaded by Visual C++.

Provides a place for you to specify PREP Phase I options. Available for function coverage, function timing, and line coverage.

Click this button to browse your directory structure for profiler batch files.

Provides a place for you to specify a batch file. A sample entry might be c:\profile.bat.

Select this option to record whether a function is called. This lets you determine which sections of your code are being executed.

Select this option to record how many times a function is called as well as how much time was spent in each function and in called functions.

Select this option to record whether a line of code was executed.

Select this option to merge the results of this profiling run with the results of the previous profiling run. The previous profiling run must have the same profile type as the current profile run (for example, line coverage or function timing).

Specifies a list of previously-issued profiler batch files.

AppHelp topics start here

Project Folder Properties: General

This property page appears in FileView when you right-click on the general name of a group of files.

Project File

Displays the name of the project.

Last Modified

Displays the date on which the project file was last changed.

Status

Indicates whether the project file is checked into or out of Microsoft Visual SourceSafe. If the file is not under source-code control, this field is blank.

Project Folder Properties: General

A read-only property page that displays the filename and last modified date for the output file.

Folder Properties: General

Set the name for a group of files in a project.

Folder name

Displays the name of the group.

File extensions

Filters files into a group when they are inserted into the project. For example, if you set the **File extensions** field to bmp;gif;jpg for the **Resource Files** folder, and then add files with those extensions to the project, the files will be grouped within in the **Resource Files** folder.

Note: You can add files without the file extensions defined for a group by using either the context menu or a drag-and-drop operation.

Source File Properties: General

File name

Displays the fully qualified name of the file.

Last modified date

Displays the date on which the file was last modified.

Persist as

Displays how the file is saved in the project. You can type additional information, such as an environment variable, into this field. The information you type must resolve to match the information in the **File name** field.

Status

Shows the source-code control status, if the file is under source-code control in a source-code control system that conforms to the Microsoft Source Code Control API Specification.

Header File Properties: General

A read-only property page.

Filename

Displays the fully qualified name of the file.

Last modified date

Displays the date on which the file was last modified.

Status

Shows the source-code control status, if the file is under source-code control in a source-code control system that conforms to the Microsoft Source Code Control API Specification.

Source File Properties: Inputs

Displays the current tools associated with the selected file.

Tools

Highlight a tool to see the associated files. These are the files connected to the selected tool.

Files

Displays the associated files for the tool shown in Tools.

Last modified date

Displays the last modified date of the file selected.

Source File Properties: Outputs

Displays the current tools associated with the selected file.

Tools

Highlight a tool to see the associated files. These are the files connected to the selected tool.

Files

Displays the associated files for the tool shown in Tools.

Last modified date

Displays the last modified date of the file selected.

Source File Properties: Dependencies

Displays the current tools associated with the selected file.

Tools

Highlight a tool to see the associated files. These are the files connected to the selected tool.

Files

Displays the associated files for the tool shown in Tools.

Last modified date

Displays the last modified date of the file selected.

Enter the command line you want to run when the user selects "Build" from the Build menu. This syntax is then displayed in the "Build command line" field on the General tab of the Project Settings dialog box, for the Debug build configuration.

Enter the name of the file that will contain the output for the command line.

Enter the command-line parameter you want added to the Build command when the user selects "Rebuild" from the Build menu. This syntax is then displayed in the "Rebuild all options" field on the General tab of the Project Settings dialog box, for the Debug build configuration.

Enter the command line you want to run when the user selects "Build" from the Build menu. This syntax is then displayed in the "Build command line" field on the General tab of the Project Settings dialog box, for the Release build.

Enter the name of the file that will contain the output for the command line.

Enter the command-line parameter you want added to the Build command when the user selects "Rebuild" from the Build menu. This syntax is then displayed in the "Rebuild all options" field on the General tab of the Project Settings dialog box, for the Release build configuration.

This option creates a project (.dsp) file with the specified name but adds no files to it. Use this when you intend to supply all your own source files.

Creates a very simple Windows application that runs and exits immediately. The application files include a <projectname>.cpp file that contains the **WinMain** function, and stdafx.cpp and stdafx.h files that are used to create the precompiled header file.

Creates a simple Windows-based "Hello, World!" application with an initial File and Help menu, minimize and maximize buttons, and a system menu. The application files include a .cpp file, precompiled headers and resource files.

Closes the ATL Object Wizard without saving any changes you have made.

Applies the properties you have selected to the object, saves the appropriate data and files, and closes the ATL Object Wizard.

Displays the property pages for the object you have selected so that you can set its properties.

Lists the ATL object categories. Each category contains different objects that can be inserted into your ATL project. When you choose a category, the icons of the ATL objects in that category are displayed on the right.

Displays the icons of the ATL objects for each category. Choose the category from the list on the left, and the icons of the ATL objects in that category are displayed on the right. Double-click an icon to insert that object into your project. **ATL Controls** category: **ATL Controls** category: **Full Control** adds an object that supports the interfaces for all containers, **Property Page** adds an object that implements a property page, **HTML Control** adds a control that includes a DHTML resource and displays an HTML web page in its user interface, **Composite Control** adds a control that can host multiple controls within itself, **Lite Control, Lite HTML Control, and Lite Composite Control** each add an object like its regular version, but with support for only the interfaces needed by Internet Explorer, including support for a user interface. **ATL Miscellaneous** category: **Dialog** adds a class that implements a dialog. **ATL Objects** category: **Simple Object** adds a minimal COM object, **Add-in Object** adds a COM object to extend the Visual C++ shell with your own command, **Internet Explorer Object** adds an object that supports the interfaces needed by Internet Explorer, but without support for a user interface, **Active Server Component** adds an object that can be used by the Active Server Pages feature of the Internet Information Server (IIS), **MS Transaction Server Object** includes the header files needed by the Transaction Server and declares the object non-aggregatable, and **Component Registrar Object** adds an object that implements the **IComponentRegistrar** interface, which can be used with the **DECLARE_OBJECT_DESCRIPTION** macro to register objects in your DLL server individually.

The control name. By default, this name becomes the root for all other names in this dialog box. However, you can enter your own names rather than accept these defaults.

The following Ids are created at compile time and do not appear in any dialog in the .RC file.

The name of the class implementing the object.

Provides a space for you to type the name of the file in which the new class(es) will be defined.

The name of the CPP file implementing the object.

The name of the component class that contains a list of interfaces supported by the object.

The interface you create for your object. This interface contains your custom methods. For ATL Controls, Internet Explorer Controls, Simple Objects, Internet Explorer Objects, Add-in Objects, Active Server Components, and MS Transaction Server Components, the wizard creates an interface with the name you specify. For Property Page objects, no custom interface is created, and the wizard assigns **IUnknown** as the object interface. For Component Registrar objects, no custom interface is created, and the wizard assigns **IComponentRegistrar** as the object interface. Dialog objects do not create an interface.

A string that describes the object and goes into the registry.

A name that containers can use instead of the object's CLSID.

Specifies that the threading model used by the object is single-threaded. Each process is a single thread of execution.

Specifies that the threading model used by the object is apartment-threaded. There can be multiple objects on a thread, but an object can reside on only one thread.

Specifies that the object supports both apartment-threaded and free-threaded threading models.

Specifies that the threading model used by the object is free-threaded. The object can be used on any thread at any time.

Creates a free-threaded marshaler object to efficiently marshal interface pointers between threads in the same process.

Specifies that the object supports a dual interface (its vtable has custom interface functions plus late-binding **IDispatch** methods). Allows both COM clients and Automation controllers to access the object.

Specifies that the object supports a custom interface (its vtable has custom interface functions). A custom interface can be faster than a dual interface, especially across process boundaries.

Specifies that the object supports the **IDispatch** interface.

Specifies that the object can be aggregated.

Specifies that the object cannot be aggregated.

Specifies that the object must be aggregated.

Creates support for the **ISupportErrorInfo** interface so your object can do error handling.

Enables connection points for your object by making your object's class derive from **IConnectionPointContainerImpl**.

Select this option to make your control completely opaque, so that none of the container shows behind the control boundaries. This helps the container draw the control more quickly. The entire control rectangle is passed to the OnDraw method. If this option is not selected, the control can contain transparent parts. This option sets the **VIEWSTATUS_OPAQUE** bit in the **VIEWSTATUS** enumeration.

Select this option to make the control's background a solid color and not a pattern. This option is meaningful only if the Opaque is option is also selected. This option sets the **VIEWSTATUS_SOLIDBKGND** bit in the **VIEWSTATUS** enumeration.

Select this option to make your control invisible at run time. You can use invisible controls to perform operations in the background, such as firing events at timed intervals.

Select this option to have your control appear in the **Insert Object** dialog box of applications such as Word and Excel. Your control can then be inserted by any application that supports embedded objects through this dialog box.

Enables your control to act like a button. Used by containers that provide default buttons. If the container has marked the control's client site as a default button, selecting this option enables your button control to display itself as a default button by drawing itself with a thicker frame. See **CComControl::GetAmbientDisplayAsDefault**. This option sets the **OLEMISC_ACTSLIKEBUTTON MiscStatus** bit.

Select this option to enable your control to replace the container's native label. This option sets the **OLEMISC_ACTSLIKELABEL MiscStatus** bit. The container is responsible for determining what to do with this flag, if anything.

Superclasses one of the predefined window classes in the drop-down list. If you choose one of the window classes from the drop-down list, when you call **Create** the window of your control will implement that window.

Select this option to have your control create a normalized device context when it is called to draw itself. This standardizes the control's appearance, but makes drawing less efficient.

Specifies that your control cannot be windowless. If you do not select this option, your control will automatically be windowless in containers that support windowless objects, and automatically be windowed in containers that do not support windowless objects. Selecting this option forces your control to be windowed, even in containers that support windowless objects.

The text that appears on the tab for this property page.

A text string describing the page. Can be displayed in the property sheet dialog box. The property frame could use the description in a status line or tool tip. The standard property frame currently does not use this string.

The name of the help file that describes how to use the property page. This name should not include a path. When the user presses **Help**, the frame opens the help file in the directory named in the value of the HelpDir key in the property page registry entries under its CLSID.

A list of the stock properties your control does not support. You can select a property and move it to the list of supported properties by clicking the right arrow button. You can move all unsupported properties to the **Supported** list by clicking the button labeled **>>**.

A list of the stock properties your control supports. You can select a property and move it to the list of unsupported properties by clicking the left arrow button. You can move all supported properties to the **Not supported** list by clicking the button labeled **<<**.

Moves the selected stock property from the **Not supported** list to the **Supported** list.

Moves the selected stock property from the **Supported** list to the **Not supported** list.

Moves all stock properties from the **Not supported** list to the **Supported** list.

Moves all stock properties from the **Supported** list to the **Not supported** list.

Adds the **OnStartPage** and **OnEndPage** methods to your object. This option must be selected to enable the Active Server Pages' Intrinsic Objects. By default, it is selected.

Provides access to the Active Server Pages' intrinsic Request object. The **OnStartPage/OnEndPage** option must be selected to enable the Active Server Pages' Intrinsic Objects.

Provides access to the Active Server Pages' intrinsic Response object. The **OnStartPage/OnEndPage** option must be selected to enable the Active Server Pages' Intrinsic Objects.

Provides access to the Active Server Pages' intrinsic Server object. The **OnStartPage/OnEndPage** option must be selected to enable the Active Server Pages' Intrinsic Objects.

Provides access to the Active Server Pages' intrinsic Session object. The **OnStartPage/OnEndPage** option must be selected to enable the Active Server Pages' Intrinsic Objects.

Provides access to the Active Server Pages' intrinsic Application object. The **OnStartPage/OnEndPage** option must be selected to enable the Active Server Pages' Intrinsic Objects.

Allows your Add-in object to catch application events.

Allows your Add-in object to catch debugger events.

Creates a toolbar button the user can click to carry out a command added by your Add-in object.

The name of the command added to the Visual C++ shell by your Add-in object. This is the name of the DLL created, the root of the project file names, and the name that appears on the **Add-ins and Macro Files** tab of the **Tools/Customize** menu option.

The name of the method that implements the command. This method is added to **ICommand** and specifies what the command does.

The text you want to appear on the button you add to the toolbar to carry out your command.

The text you want to appear on the status line when your command is executing.

The tooltip message you want to appear when the user's mouse hovers over your toolbar button.

Provides your object access to the three **IObjectControl** methods: **Activate**, **CanBePooled**, and **Deactivate**.

Tells the Transaction Server run-time enviroment that your object should be returned to an instance pool after deactivation, rather than destroyed. This option cannot be selected unless the **IObjectControl** option is also selected.

ATL Objects category: **Simple Object** adds a minimal COM object, **Add-in Object** adds a COM object to extend the Visual C++ shell with your own command, **Internet Explorer Object** adds an object that supports the interfaces needed by Internet Explorer, but without support for a user interface, **Active Server Component** adds an object that can be used by the Active Server Pages feature of the Internet Information Server (IIS), **MMC Snapin** adds a snapin for Microsoft Management Console that is part of Microsoft Backoffice, **MS Transaction Server Object** includes the header files needed by the Transaction Server and declares the object non-aggregatable, and **Component Registrar Object** adds an object that implements the **IComponentRegistrar** interface, which can be used with the **DECLARE_OBJECT_DESCRIPTION** macro to register objects in your DLL server individually.

ATL Controls category: **Full Control** adds an object that supports the interfaces for all containers, **Property Page** adds an object that implements a property page, **HTML Control** adds a control that includes a DHTML resource and displays an HTML web page in its user interface, **Composite Control** adds a control that can host multiple controls within itself, **Lite Control, Lite HTML Control, and Lite Composite Control** each add an object like its regular version, but with support for only the interfaces needed by Internet Explorer, including support for a user interface.

ATL Miscellaneous category: **Dialog** adds a class that implements a dialog.

ATL Data Access category: **OLE DB Provider** and **Consumer** add code derived from OLE DB Templates to provide OLE DB provider and consumer support

Using the description for IDD_ATTRIBUTES

Using the description provided in IDD_ATTRIBUTES

Provides a space for you to type the class name that will be created for the accessor.

Specifies whether the control will support the ability to change the returned data.

Provides a space for you to type the class name that will be created for the command or table and will contain the Open method to open the datasource.

Specifies to use CCommand to create the command.

Specifies whether you want to support the ability to delete existing records.

Provides the space for you to type the name of the file in which the new class(es) will be defined.

Specifies whether you want to support the ability to insert new records.

Provides a space for you to type the name that is used to generate a default name for the Class and Accessor.

Specifies that you want to use CTable to open the query.

Click this to select a datasource for the control.

Defines names to be used in the OLE DB provider.

The name of the component class that contains a list of interfaces supported by the object.

A name that the provider can use instead of the object's CLSID.

The name you want to give to the OLE DB command object.

The name you want to give to the file that implements the OLE DB command object.

The name you want to give to the OLE DB data source object.

The name you want to give to the file that implements the OLE DB data source object.

The name you want to give to the OLE DB rowset object.

The name you want to give to the file that implements the OLE DB rowset object.

The name you want to give to the OLE DB session object.

The name you want to give to the file that implements the OLE DB session object.

A version number used with the ProgID and CoClass to generate a version-dependent programmatic ID.

The control name. By default, this name becomes the root for all other names in this dialog box. However, you can enter your own names rather than accept these defaults.

Check if the Snap-In node will be an extension Snap-In node.

HThe name of an existing Snap-In node that acts as the parent of the extended Snap-In node.

Invokes the Snap-In dialog box.

Implements a persistence interface for the Snap-In node.

Implements the loading and saving of a Snap-In node.

Implements initialized stream-based persistence of a Snap-In node.

Allows MMC Console to get the version and copyright information for the Snap-In node.

Implements the IComponent interface for the Snap-In node.

Implements the IComponentData interface for the Snap-In node.

Allows extending of the context menu for the Snap-In node.

Allows extending of the control bar for the Snap-In node.

Allows extending of the property sheet for the Snap-In node.

Implements stream-based persistence of a Snap-In node.

List of available Snap-In nodes for the system.

Select the table or procedure for which you want to generate the OLE DB accessor class.

Opens a dialog you can use to browse to the location of the type library you wish to select.

Select the type library (or type libraries) to use by clicking the box next to your selection to mark it. To deselect the type library, click the box to clear it.

Lists the full name and the version number of the currently highlighted type library.

Lists the path and file name of the currently-highlighted type library.

Click to add a type library to the project.

Indicates a type library that you have added to the object.

Opens a dialog you can use to browse to the location of the file into which your proxy code will be written.

The name of the file where the proxy code will be generated. By default, this file is new, and its name is based on the type library name, appended with “CP”.

Lists the interfaces exposed in the type library named on the tab of the currently active tab control. To implement an interface, click the box next to the interface to mark it. To deselect, click the box again to clear it.

Add ATL Support to Your MFC Project?

Specify whether you want to add the required support for the Active Template Library to your MFC-based project. In order to add any of the ATL objects or controls to your MFC project, you must first add the ATL support.

Yes
The ATL Object Wizard inserts code that supports the Active Template Library, and then displays the ATL Object Wizard dialog box, which you can use to specify which object or control you wish to add to your project.

For details about the inserted code, see the Remarks section. For DLL projects, also see Notes for DLL Projects.

No
Does not add code support for ATL to the project.

Remarks

If you click **OK**, Visual C++ adds the following code to your MFC-based project:

- An .idl file which contains information to create an initial type library.

  Note For MFC automation-enabled projects, ATL uses the existing .odl file.

- An .rgs file which contains the registry script to register the server.
- An object map.
- A typelib entry, and, to properly register the server, an entry for the new .rgs file. Both of these are added to the project's resource file.
- A new **CComModule**-derived class, with several methods that initialize and terminate the ATL support of the project.

Visual C++ also:

- Adds a new function to theCWinApp-derived class (called from **InitInstance)** to initialize ATL. Code is also added to the **ExitInstance** function for ATL termination.
- Includes new header files (<project_name>_i.h and <project_name>_i.c) in the implementation file, declaring and initializing the new GUIDs for the **CComModule**-derived class.
- In the STDAFX.h and STDAFX.cpp files, declares and defines the **CComModule**-derived class, and includes new files (STATREG.h and ATLIMPL.h) for static registry of the ATL support.

Notes for DLL Projects

When adding ATL support to an MFC DLL project, there are some notable differences. First, a global variable, _Module, of type **CComModule** is added. In addition, code is added to the **DLLRegisterServer** and **DLLUnregisterServer** functions for registering and unregistering the DLL.

Note The ATL Object Wizard does not add entries to the DEF file for an MFC DLL. You must do this by hand.

If your project is a DLL COM server, export four functions by adding the following to the EXPORTS section of your project's .def file:

```
DllCanUnloadNow    @1 PRIVATE
DllGetClassObject    @2 PRIVATE
DllRegisterServer    @3 PRIVATE
DllUnregisterServer    @4 PRIVATE
```

The .idl file initially contains the following code:

```
    // MFC_Gen.idl : IDL source for MFC_Gen.exe
//
// This file will be processed by the MIDL tool to
// produce the type library (MFC_Gen.tlb) and marshalling code.
import "oaidl.idl";
import "ocidl.idl";
```

```
[
    uuid(D2051A26-5D3B-11D1-A371-00A0C90F2851),
    version(1.0),
    helpstring("MFC_Gen 1.0 Type Library")
]
library MFC_GenLib
{
    importlib("stdole32.tlb");
    importlib("stdole2.tlb");
};
```

The .rgs file initially contains the following code:

```
{
    NoRemove AppID
    {
        {D2051A27-5D3B-11D1-A371-00A0C90F2851} = s 'MFC_Gen'
        'MFC_Gen.EXE'
        {
            val AppID = s {D2051A27-5D3B-11D1-A371-00A0C90F2851}
        }
    }
}
```

(Note that the GUID is just an example.)

Here is the object map generated for a Simple ATL object called MySimple:

```
CMFC_Gen2Module _Module;

BEGIN_OBJECT_MAP(ObjectMap)
    OBJECT_ENTRY(CLSID_MySimple, CMySimple)
END_OBJECT_MAP()
```

A name that will be exported as a Win32 function. Extended stored procedure names commonly begin with **xp_**.

Enablesyou to choose the language in which your documentation is displayed, if more than one language is available with your MSDN installation. Select from the drop-down list.

Displays a list of the documentation collections installed on your machine from which you can choose the one you prefer to use. For example, newer installations may include updated versions of topics; or, you might customize a collection with preferences such as bookmarks, search subsets and so forth. The <Default> collection is the newest collection, based on the timestamp of the collection.