Add Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthAddinAddC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthAddinAddX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthAddinAddA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthAddinAddS"}

Adds an object to a collection.

Syntax

object.**Add(***component***)**

The **Add** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
component	Required. For the **LinkedWindows** collection, an object. For the **VBComponents** collection, an enumerated constant representing a class module, a form, or a standard module.

You can use one of the following constants for the *component* argument:

Constant	Description
vbext_ct_ClassModule	Adds a class module to the collection.
vbext_ct_MSForm	Adds a form to the collection.
vbext_ct_StdModule	Adds a standard module to the collection.

Remarks

For the **LinkedWindows** collection, the **Add** method adds a window to the collection of currently linked windows.

Note You can add a window that is a pane in one linked window frame to another linked window frame; the window is simply moved from one pane to the other. If the linked window frame that the window was moved from no longer contains any panes, it's destroyed.

For the **VBComponents** collection, the **Add** method creates a new standard component and adds it to the project.

For the **VBComponents** collection, the **Add** method returns a **VBComponent** object. For the **LinkedWindows** collection, the **Add** method returns **Nothing**.

AddFromFile Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthAddFromFileC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthAddFromFileX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthAddFromFileA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthAddFromFileS"}

For the **References** collection, adds a reference to a project from a file. For the **CodeModule** object, adds the contents of a file to a module.

Syntax

object.**AddFromFile(***filename***)**

The **AddFromFile** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
filename	Required. A string expression specifying the name of the file you want to add to the project or module. If the file name isn't found and a path name isn't specified, the directories searched by the **Windows OpenFile** function are searched.

Remarks

For the **CodeModule** object, the **AddFromFile** method inserts the contents of the file starting on the line preceding the first procedure in the code module. If the module doesn't contain procedures, **AddFromFile** places the contents of the file at the end of the module.

AddFromGuid Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthAddFromGuidC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthAddFromGuidX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthAddFromGuidA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthAddFromGuidS"}

Adds a reference to the **References** collection using the globally unique identifier (GUID) of the reference.

Syntax

object.**AddFromGuid(***guid***,** *major***,** *minor***) As Reference**

The **AddFromGuid** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
guid	Required. A string expression representing the GUID of the reference.
major	Required. A **Long** specifying the major version number of the reference.
minor	Required. A **Long** specifying the minor version number of the reference.

Remarks

The **AddFromGuid** method searches the registry to find the reference you want to add. The GUID can be a type library, control, class identifier, and so on.

AddFromString Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthAddFromStringC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthAddFromStringX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthAddFromStringA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthAddFromStringS"}

Adds text to a module.

Syntax

object.**AddFromString**

The *object* placeholder is an object expression that evaluates to an object in the Applies To list.

Remarks

The **AddFromString** method inserts the text starting on the line preceding the first procedure in the module. If the module doesn't contain procedures, **AddFromString** places the text at the end of the module.

Close Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthCloseC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthCloseX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthCloseS"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthCloseA"}

Closes and destroys a window.

Syntax

object.**Close**

The *object* placeholder is an object expression that evaluates to an object in the Applies To list.

Remarks

The following types of windows respond to the **Close** method in different ways:

- For a window that is a code pane, **Close** destroys the code pane.
- For a window that is a designer, **Close** destroys the contained designer.
- For windows that are always available on the **View** menu, **Close** hides the window.

CreateEventProc Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthCreateEventProcC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthCreateEventProcX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthCreateEventProcA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthCreateEventProcS"}

Creates an event procedure.

Syntax

object.**CreateEventProc(***eventname*, *objectname*) **As Long**

The **CreateEventProc** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
eventname	Required. A string expression specifying the name of the event you want to add to the module.
objectname	Required. A string expression specifying the name of the object that is the source of the event.

Remarks

Use the **CreateEventProc** method to create an event procedure. For example, to create an event procedure for the Click event of a **Command Button** control named `Command1` you would use the following code, where `CM` represents a object of type **CodeModule**:

```
TextLocation = CM.CreateEventProc("Click", "Command1")
```

The **CreateEventProc** method returns the line at which the body of the event procedure starts. **CreateEventProc** fails if the arguments refer to a nonexistent event.

DeleteLines Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthDeleteLinesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthDeleteLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthDeleteLinesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthDeleteLinesS"}

Deletes a single line or a specified range of lines.

Syntax

object.**DeleteLines (***startline* [, *count*]**)**

The **DeleteLines** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
startline	Required. A **Long** specifying the first line you want to delete.
count	Optional. A **Long** specifying the number of lines you want to delete.

Remarks

If you don't specify how many lines you want to delete, **DeleteLines** deletes one line.

Export Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthExportC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthExportX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthExportA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthExportS"}

Saves a component as a separate file or files.

Syntax

object.**Export(***filename***)**

The **Export** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
filename	Required. A **String** specifying the name of the file that you want to export the component to.

Remarks

When you use the **Export** method to save a component as a separate file or files, use a file name that doesn't already exist; otherwise, an error occurs.

Find Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthFindC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthFindX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthFindS"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthFindA"}

Searches the active module for a specified string.

Syntax

object.**Find(***target***,** *startline***,** *startcol***,** *endline***,** *endcol* [, *wholeword*] [, *matchcase*] [, *patternsearch*]**) As Boolean**

The **Find** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
target	Required. A **String** containing the text or pattern you want to find.
startline	Required. A **Long** specifying the line at which you want to start the search; will be set to the line of the match if one is found.
startcol	Required. A **Long** specifying the column at which you want to start the search; will be set to the column containing the match if one is found.
endline	Required. A **Long** specifying the last line of the match if one is found.
endcol	Required. A **Long** specifying the last line of the match if one is found.
wholeword	Optional. A **Boolean** value specifying whether to only match whole words. If **True**, only matches whole words. **False** is the default.
matchcase	Optional. A **Boolean** value specifying whether to match case. If **True**, the search is case sensitive. **False** is the default.
patternsearch	Optional. A **Boolean** value specifying whether or not the target string is a regular expression pattern. If **True**, the target string is a regular expression pattern. **False** is the default.

Remarks

Find returns **True** if a match is found and **False** if a match isn't found.

The *matchcase* and *patternmatch* arguments are mutually exclusive; if both arguments are passed as **True**, an error occurs.

The content of the **Find** dialog box isn't affected by the **Find** method.

GetSelection Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthGetSelectionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthGetSelectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthGetSelectionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthGetSelectionS"}

Returns the selection in a code pane.

Syntax

object.**GetSelection(***startline*, *startcol*, *endline*, *endcol***)**

The **GetSelection** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
startline	Required. A **Long** that returns a value specifying the first line of the selection in the code pane.
startcol	Required. A **Long** that returns a value specifying the first column of the selection in the code pane.
endline	Required. A **Long** that returns a value specifying the last line of the selection in the code pane.
endcol	Required. A **Long** that returns a value specifying the last column of the selection in the code pane.

Remarks

When you use the **GetSelection** method, information is returned in output arguments. As a result, you must pass in variables because the variables will be modified to contain the information when returned.

Import Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthImportC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthImportX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthImportA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthImportS"}

Adds a component to a project from a file; returns the newly added component.

Syntax

object.**Import(***filename***) As VBComponent**

The **Import** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
filename	Required. A **String** specifying path and file name of the component that you want to import the component from.

Remarks

You can use the **Import** method to add a component, form, module, class, and so on, to your project.

InsertLines Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthInsertLinesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthInsertLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthInsertLinesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthInsertLinesS"}

Inserts a line or lines of code at a specified location in a block of code.

Syntax

object.**InsertLines**(*line*, *code*)

The **InsertLines** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
line	Required. A **Long** specifying the location at which you want to insert the code.
code	Required. A **String** containing the code you want to insert.

Remarks

If the text you insert using the **InsertLines** method is carriage return–linefeed delimited, it will be inserted as consecutive lines.

Item Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthAddinItemC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthAddinItemX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthAddinItemA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthAddinItemS"}

Returns the indexed member of a collection.

Syntax

object.**Item(***index***)**

The **Item** method syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
index	Required. An expression that specifies the position of a member of the collection. If a numeric expression, *index* must be a number from 1 to the value of the collection's **Count** property. If a string expression, *index* must correspond to the ***key*** argument specified when the member was added to the collection.

The following table lists the collections and their corresponding ***key*** arguments for use with the **Item** method. The string you pass to the **Item** method must match the collection's ***key*** argument.

Collection	Key argument
Windows	**Caption** property setting
LinkedWindows	**Caption** property setting
CodePanes	No unique string is associated with this collection.
VBProjects	**Name** property setting
VBComponents	**Name** property setting
References	**Name** property setting
Properties	**Name** property setting

Remarks

The *index* argument can be a numeric value or a string containing the title of the object.

Lines Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthLinesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthLinesS"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthLinesA"}

Returns a specified line of code.

Syntax

object.**Lines(***startline*, *count***) As String**

The **Lines** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
startline	Required. A **Long** specifying the first line of the code you want to return.
count	Required. A **Long** specifying the number of lines you want to return.

Remarks

The line numbers in a code module begin at 1.

ProcBodyLine Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthProcBodyLineC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthProcBodyLineX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthProcBodyLineA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthProcBodyLineS"}

Returns the first line of a procedure.

Syntax

object.**ProcBodyLine(***procname***,** *prockind***) As Long**

The **ProcBodyLine** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
procname	Required. A **String** containing the name of the procedure.
prockind	Required. Specifies the kind of procedure to locate. Because property procedures can have multiple representations in the module, you must specify the kind of procedure you want to locate. All procedures other than property procedures (that is, **Sub** and **Function** procedures) use **vbext_pk_Proc**.

You can use one of the following constants for the *prockind* argument:

Constant	Description
vbext_pk_Get	Specifies a procedure that returns the value of a property.
vbext_pk_Let	Specifies a procedure that assigns a value to a property.
vbext_pk_Set	Specifies a procedure that sets a reference to an object.
vbext_pk_Proc	Specifies all procedures other than property procedures.

Remarks

The first line of a procedure is the line on which the **Sub**, **Function**, or **Property** statement appears.

ProcCountLines Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthProcCountLinesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthProcCountLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthProcCountLinesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthProcCountLinesS"}

Returns the number of lines in the specified procedure.

Syntax

object.**ProcCountLines(***procname***,** *prockind***) As Long**

The **ProcCountLines** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
procname	Required. A **String** containing the name of the procedure.
prockind	Required. Specifies the kind of procedure to locate. Because property procedures can have multiple representations in the module, you must specify the kind of procedure you want to locate. All procedures other than property procedures (that is, **Sub** and **Function** procedures) use **vbext_pk_Proc**.

You can use one of the following constants for the *prockind* argument:

Constant	Description
vbext_pk_Get	Specifies a procedure that returns the value of a property.
vbext_pk_Let	Specifies a procedure that assigns a value to a property.
vbext_pk_Set	Specifies a procedure that sets a reference to an object.
vbext_pk_Proc	Specifies all procedures other than property procedures.

Remarks

The **ProcCountLines** method returns the count of all blank or comment lines preceding the procedure declaration and, if the procedure is the last procedure in a code module, any blank lines following the procedure.

ProcOfLine Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthProcOfLineC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthProcOfLineX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthProcOfLineA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthProcOfLineS"}

Returns the name of the procedure that the specified line is in.

Syntax

object.**ProcOfLine**(*line*, *prockind*) **As String**

The **ProcOfLine** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
line	Required. A **Long** specifying the line to check.
prockind	Required. Specifies the kind of procedure to locate. Because property procedures can have multiple representations in the module, you must specify the kind of procedure you want to locate. All procedures other than property procedures (that is, **Sub** and **Function** procedures) use **vbext_pk_Proc**.

You can use one of the following constants for the *prockind* argument:

Constant	Description
vbext_pk_Get	Specifies a procedure that returns the value of a property.
vbext_pk_Let	Specifies a procedure that assigns a value to a property.
vbext_pk_Set	Specifies a procedure that sets a reference to an object.
vbext_pk_Proc	Specifies all procedures other than property procedures.

Remarks

A line is within a procedure if it's a blank line or comment line preceding the procedure declaration and, if the procedure is the last procedure in a code module, a blank line or lines following the procedure.

ProcStartLine Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthProcStartLineC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthProcStartLineX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthProcStartLineA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthProcStartLineS"}

Returns the line at which the specified procedure begins.

Syntax

object.**ProcStartLine(***procname***,** *prockind***) As Long**

The **ProcStartLine** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
procname	Required. A **String** containing the name of the procedure.
prockind	Required. Specifies the kind of procedure to locate. Because property procedures can have multiple representations in the module, you must specify the kind of procedure you want to locate. All procedures other than property procedures (that is, **Sub** and **Function** procedures) use **vbext_pk_Proc**.

You can use one of the following constants for the *prockind* argument:

Constant	Description
vbext_pk_Get	Specifies a procedure that returns the value of a property.
vbext_pk_Let	Specifies a procedure that assigns a value to a property.
vbext_pk_Set	Specifies a procedure that sets a reference to an object.
vbext_pk_Proc	Specifies all procedures other than property procedures.

Remarks

A procedure starts at the first line below the **End Sub** statement of the preceding procedure. If the procedure is the first procedure, it starts at the end of the general Declarations section.

Remove Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthAddinRemoveC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthAddinRemoveX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthAddinRemoveA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthAddinRemoveS"}

Removes an item from a collection.

Syntax

object.**Remove(***component***)**

The **Remove** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
component	Required. For the **LinkedWindows** collection, an object. For the **References** collection, a reference to a type library or a project. For the **VBComponents** collection, an enumerated constant representing a class module, a form, or a standard module.

Remarks

When used on the **LinkedWindows** collection, the **Remove** method removes a window from the collection of currently linked windows. The removed window becomes a floating window that has its own linked window frame.

ReplaceLine Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthReplaceLineC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthReplaceLineX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthReplaceLineA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthReplaceLineS"}

Replaces an existing line of code with a specified line of code.

Syntax

object.**ReplaceLine**(*line*, *code*)

The **ReplaceLine** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
line	Required. A **Long** specifying the location of the line you want to replace.
code	Required. A **String** containing the code you want to insert.

SetFocus Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthSetFocusC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthSetFocusX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthSetFocusA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthSetFocusS"}

Moves the focus to the specified window.

Syntax

object.**SetFocus**

The *object* placeholder is an object expression that evaluates to an object in the Applies To list.

Remarks

Use the **SetFocus** method on windows that are already visible.

SetSelection Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthSetSelectionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthSetSelectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthSetSelectionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthSetSelectionS"}

Sets the selection in the code pane.

Syntax

object.**SetSelection(***startline*, *startcol*, *endline*, *endcol***)**

The **SetSelection** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
startline	Required. A **Long** specifying the first line of the selection.
startcol	Required. A **Long** specifying the first column of the selection.
endline	Required. A **Long** specifying the last line of the selection.
endcol	Required. A **Long** specifying the last column of the selection.

Show Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthShowC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vamthShowX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vamthShowA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vamthShowS"}

Makes the specified code pane the visible code pane in its window.

Syntax

object.**Show**

The *object* placeholder is an object expression that evaluates to an object in the Applies To list.

Remarks

The **Show** method makes the specified code pane the pane with the focus in its window.

Add Method Example

The following example uses the **Add** method to add one standard module to the **VBComponents** collection.

```
Application.VBE.VBProjects(1).VBComponents.Add(vbext_ct_StdModule)
```

AddFromFile Method Example

The following example uses the **AddFromFile** method to add the contents of a file to a specified code pane.

```
Application.VBE.CodePanes(3).CodeModule.AddFromFile "c:\Code Files\
book2.frm"
```

AddFromGUID Method Example

The following example uses the **AddFromGUID** method to add a reference to the current project, identifying the reference using the globally unique ID value of the **Reference** object.

```
Application.VBE.ActiveVBProject.References.AddFromGuid("{000204EF-0000-
0000-C000-000000000046}", 5, 0)
```

AddFromString Method Example

The following example uses the **AddFromString** method to add a line, “Dim intJack As Integer,” to the specified code pane.

```
Application.VBE.CodePanes(3).CodeModule.AddFromString "Dim intJack As
Integer"
```

Close Method Example

The following example uses the **Close** method to close a specified member of the **Windows** collection.

```
Application.VBE.Windows(9).Close
```

CreateEventProc Method Example

The following example uses the **CreateEventProc** method to create the Button_Click procedure.

```
Debug.Print
Application.VBE.SelectVBComponents.CodeModule.CreateEventProc("Click",
"Button")
```

DeleteLines Method Example

The following example has two steps. The first **For…Next** loop uses the **InsertLines** method to insert into `CodePanes(1)` 26 ever-longer initial segments of the alphabet, starting with “a.” The last line inserted is the entire alphabet.

The second **For…Next** loop uses the **DeleteLines** method to delete the odd-numbered lines. Although it seems that the second loop should simply delete every other line, note that after each deletion the lines get renumbered. Therefore the deletion is advancing by two lines at each step, one line because I is increasing by one and another line because the larger line numbers are each decreasing by one.

```
For I = 1 to 26
   Application.VBE.SelectedVBComponent.CodeModule.InsertLines i, Mid$
("abcdefghijklmnopqrstuvwxyz", 1, I)
Next
For I = 1 to 13
   Application.VBE.SelectedVBComponent.CodeModule.DeleteLines I
Next
```

Export Method Example

The following example creates a file named `test.bas` and uses the **Export** method to copy the contents of the `VBComponents(1)` code module into the file.

```
Application.VBE.ActiveVBProject.VBComponents(1).Export("test.bas")
```

Find Method Example

The following example uses the **Find** method to verify that the specified block of lines, lines 1261 through 1279, of a particular code pane does contain the string “Tabs.Clear.”

```
Application.VBE.CodePanes(2).CodeModule.Find ("Tabs.Clear", 1261, 1, 1280,
1, False, False)
```

GetSelection Method Example

The following example returns the locations of the starting and ending points of the current selection in `CodePanes(1)`. The last line in the example uses the **GetSelection** method to place the four values in the four variables.

```
Dim m As Long
Dim n As Long
Dim x As Long
Dim y As Long
Application.VBE.CodePanes(1).GetSelection m, n, x, y
```

Import Method Example

The following example uses the **Import** method on the **VBComponents** collection to copy the contents of the test.bas file into the a code module.

```
Application.VBE.ActiveVBProject.VBComponents.Import("test.bas")
```

InsertLines Method Example

The following example uses the **InsertLines** method to insert a line, “Option Explicit,” in the specified code pane.

```
Application.VBE.CodePanes(1).CodeModule.InsertLines 1, "Option Explicit"
```

Item Method Example

The following example contains two ways to display a specific member of the **CodePanes** collection, one using the **Item** method.

```
Application.VBE.CodePanes.Item(2).Show
Application.VBE.CodePanes(2).Show
```

Lines Method Example

The following example uses the **Lines** method to return a specific block of code, lines 1 through 4, in a particular code pane.

```
Debug.Print Application.VBE.CodePanes(1).CodeModule.Lines( 1, 4)
```

ProcBodyLine Method Example

The following example uses the **ProcBodyLine** method to return the line number of the first line of code in the specified procedure, SetupTabs, in a particular code pane.

```
Debug.Print Application.VBE.CodePanes(3).CodeModule.ProcBodyLine
("SetupTabs", vbext_pk_Proc)
```

ProcCountLines Method Example

The following example uses the **ProcCountLines** method to return the number of lines of code in the specified procedure, SetupTabs, in a particular code pane.

```
Debug.Print Application.VBE.CodePanes(3).CodeModule.ProcCountLines
("SetupTabs", vbext_pk_Proc)
```

ProcOfLine Method Example

The following example uses the **ProcOfLine** method to return the name of the procedure containing the specified line number in a particular code pane.

```
Debug.Print Application.VBE.CodePanes(3).CodeModule.ProcOfLine (1270,
vbext_pk_Proc)
```

ProcStartLine Method Example

The following example uses the **ProcStartLine** method to return the line at which the specified procedure begins in a particular code pane.

```
Debug.Print Application.VBE.CodePanes(3).CodeModule.ProcStartLine
("SetupTabs", vbext_pk_Proc)
```

Remove Method Example

The example verifies that a particular member of the **VBComponents** collection is a module, and then it uses the **Remove** method to remove the module.

```
Debug.Print Application.VBE.ActiveVBProject.VBComponents(4).Name
Application.VBE.ActiveVBProject.VBComponents.Remove
Application.VBE.ActiveVBProject.VBComponents(4)
```

ReplaceLine Method Example

The following example has two steps. The first **For…Next** loop uses the **InsertLines** method to insert into `CodePanes(1)` 26 ever-longer initial segments of the alphabet, starting with “a.” The last line inserted is the entire alphabet.

The second **For…Next** loop uses the **ReplaceLine** method to replace each even-numbered line with the last letter in the string that previously occupied that line. Odd-numbered lines are unchanged.

```
For I = 1 to 26
   Application.VBE.CodePanes(1).CodeModule.InsertLines I, Mid$
("abcdefghijklmnopqrstuvwxyz", 1, I)
Next I
For I = 1 to 13
   Application.VBE.CodePanes(1).CodeModule.ReplaceLine 2*I, Mid$
("abcdefghijklmnopqrstuvwxyz", 1, I)
Next I
```

SetFocus Method Example

The following example uses the **SetFocus** method to move the focus to a particular member of the **Windows** collection; that is, it makes that window behave as if you had clicked its title bar with your mouse.

```
Application.VBE.Windows(9).SetFocus
```

SetSelection Method Example

The following example uses the **SetSelection** method to select the text whose first character is the one immediately after the fourth character on the second line of `CodePanes(1)` and whose last character is the fifteenth character on the third line.

```
Application.VBE.CodePanes(1).SetSelection 2,4,3,15
```

Show Method Example

The following example uses the **Show** method to move the specified code pane to the foreground.

```
Application.VBE.CodePanes(2).Show
```

Click Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaevtClickC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaevtClickX":1}
{ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaevtClickA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaevtClickS"}

Occurs when the **OnAction** property of a corresponding command bar control is set.

Syntax

Sub *object*_**Click (ByVal** ***ctrl*** **As Object, ByRef** ***handled*** **As Boolean, ByRef** ***canceldefault*** **As Boolean)**

The Click event syntax has these ***named arguments***:

Part	Description
ctrl	Required; **Object**. Specifies the object that is the source of the Click event.
handled	Required; **Boolean**. If **True**, other add-ins should handle the event. If **False**, the action of the command bar item has not been handled.
canceldefault	Required; **Boolean**. If **True**, default behavior is performed unless canceled by a downstream add-in. If **False**, default behavior is not performed unless restored by a downstream add-in.

Remarks

The Click event is specific to the **CommandBarEvents** object. Use a variable declared using the **WithEvents** keyword to receive the Click event for a **CommandBar** control. This variable should be set to the return value of the **CommandBarEvents** property of the **Events** object. The **CommandBarEvents** property takes the **CommandBar** control as an argument. When the **CommandBar** control is clicked (for the variable you declared using the **WithEvents** keyword), the code is executed.

ItemAdded Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaevtItemAddedC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaevtItemAddedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaevtItemAddedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaevtItemAddedS"}

Occurs after a reference is added.

Syntax

Sub *object*_**ItemAdded(ByVal** *item* **As Reference)**

The required *item* argument specifies the item that was added.

Remarks

The ItemAdded event occurs when a **Reference** is added to the **References** collection.

ItemRemoved Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaevtItemRemovedC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaevtItemRemovedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaevtItemRemovedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaevtItemRemovedS"}

Occurs after a reference is removed from a project.

Syntax

Sub *object*_**ItemRemoved(ByVal** *item* **As Reference)**

The required *item* argument specifies the **Reference** that was removed.

Click Event Example

The following example illustrates how you can set up code for a Click event procedure using **WithEvents** and **Set**. Note that the object reference `ce` is used in place of the menu name **Tools** in the name of the Click event.

```
Private WithEvents ce As CommandBarEvents

Sub Test()
    Dim c As CommandBarControl
    Set c = Application.VBE.CommandBars("Tools").Controls(1)
    Set ce = Application.VBE.Events.CommandBarEvents(c)
End Sub

Private Sub ce_Click(ByVal CommandBarControl As Object, Handled As Boolean,
CancelDefault As Boolean)
    ' Put event-handling code here
End Sub
```

SelectedVBComponent Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproSelectedVBComponentC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproSelectedVBComponentX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproSelectedVBComponentA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproSelectedVBComponentS"}

Returns the selected component. Read-only.

Remarks

The **SelectedVBComponent** property returns the selected component in the **Project** window. If the selected item in the **Project** window isn't a component, **SelectedVBComponent** returns **Nothing**.

ActiveVBProject Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproActiveVBProjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproActiveVBProjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproActiveVBProjectA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproActiveVBProjectS"}

Returns the active project in the **Project** window. Read-only.

Remarks

The **ActiveVBProject** property returns the project that is selected in the **Project** window or the project in which the components are selected. In the latter case, the project itself isn't necessarily selected. Whether or not the project is explicitly selected, there is always an active project .

ActiveWindow Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproActiveWindowC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproActiveWindowX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproActiveWindowA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproActiveWindowS"}

Returns the active window in the development environment. Read-only.

Remarks

When more than one window is open in the development environment, the **ActiveWindow** property setting is the window with the focus. If the main window has the focus, **ActiveWindow** returns **Nothing**.

Caption Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproCaptionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproCaptionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproCaptionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproCaptionS"}

Returns a **String** containing the title of the active window. Read-only.

Remarks

The title of the active window is the text displayed in the window's title bar.

CodeModule Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproCodeModuleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproCodeModuleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproCodeModuleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproCodeModuleS"}

Returns an object representing the code behind the component. Read-only.

Remarks

The **CodeModule** property returns **Nothing** if the component doesn't have a code module associated with it.

Note The **CodePane** object represents a visible code window. A given component can have several **CodePane** objects. The **CodeModule** object represents the code within a component. A component can only have one **CodeModule** object.

CodePane Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproCodePaneC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproCodePaneX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproCodePaneA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproCodePaneS"}

Returns a **CodePane** object. Read-only.

Remarks

If a code pane exists, it becomes the active code pane, and the window that contains it becomes the active window. If a code pane doesn't exist for the module, the **CodePane** property creates one.

CodePaneView Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproCodePaneViewC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproCodePaneViewX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproCodePaneViewA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproCodePaneViewS"}

Returns a value indicating whether the code pane is in Procedure view or Full Module view. Read-only.

Return Values

The **CodePaneView** property return values are:

Constant	Description
vbext_cv_ProcedureView	The specified code pane is in Procedure view.
vbext_cv_FullModuleView	The specified project is in Full Module view.

Collection Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproCollectionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproCollectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproCollectionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproCollectionS"}

Returns the collection that contains the object you are working with. Read-only.

Remarks

Most objects in this object model have either a **Parent** property or a **Collection** property that points to the object's parent object.

Use the **Collection** property to access the properties, methods, and controls of the collection to which the object belongs.

CommandBarEvents Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproCommandBarEventsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproCommandBarEventsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproCommandBarEventsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproCommandBarEventsS"}

Returns the **CommandBarEvents** object. Read-only.

Settings

The setting for the argument you pass to the **CommandBarEvents** property is:

Argument	Description
vbcontrol	Must be an object of type **CommandBarControl**.

Remarks

Use the **CommandBarEvents** property to return an event source object that triggers an event when a command bar button is clicked. The argument passed to the **CommandBarEvents** property is the command bar control for which the Click event will be triggered.

Count Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproAddinCountC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproAddinCountX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproAddinCountA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproAddinCountS"}

Returns a **Long** containing the number of items in a collection. Read-only.

CountOfLines Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproCountOfLinesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproCountOfLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproCountOfLinesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproCountOfLinesS"}

Returns a **Long** containing the number of lines of code in a code module. Read-only.

CountOfDeclarationLines Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproCountOfDeclarationLinesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproCountOfDeclarationLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproCountOfDeclarationLinesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproCountOfDeclarationLinesS"}

Returns a **Long** containing the number of lines of code in the Declarations section of a code module. Read-only.

CountOfVisibleLines Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproCountOfVisibleLinesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproCountOfVisibleLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproCountOfVisibleLinesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproCountOfVisibleLinesS"}

Returns a **Long** containing the number of lines visible in a code pane. Read-only.

ActiveCodePane Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproActiveCodePaneC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproActiveCodePaneX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproActiveCodePaneA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproActiveCodePaneS"}

Returns the active or last active **CodePane** object or sets the active **CodePane** object. Read/write.

Remarks

You can set the **ActiveCodePane** property to any valid **CodePane** object, as shown in the following example:

```
Set MyApp.VBE.ActiveCodePane = MyApp.VBE.CodePanes(1)
```

The preceding example sets the first code pane in a collection of code panes to be the active code pane. You can also activate a code pane using the **Set** method.

Description Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproAddinDescriptionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproAddinDescriptionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproAddinDescriptionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproAddinDescriptionS"}

Returns or sets a string expression containing a descriptive string associated with an object. For the **VBProject** object, read/write; for the **Reference** object, read-only.

Remarks

For the **VBProject** object, the **Description** property returns or sets a descriptive string associated with the active project.

For the **Reference** object, the **Description** property returns the descriptive name of the reference.

Designer Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproDesignerC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproDesignerX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproDesignerA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproDesignerS"}

Returns the object that enables you to access the design characteristics of a component.

Remarks

If the object has an open designer, the **Designer** property returns the open designer; otherwise a new designer is created. The designer is a characteristic of certain **VBComponent** objects. For example, when you create certain types of **VBComponent** object, a designer is created along with the object. A component can have only one designer, and it's always the same designer. The **Designer** property enables you to access a component-specific object. In some cases, such as in standard modules and class modules, a designer isn't created because that type of **VBComponent** object doesn't support a designer.

The **Designer** property returns **Nothing** if the **VBComponent** object doesn't have a designer.

DesignerWindow Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproDesignerWindowC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproDesignerWindowX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproDesignerWindowA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproDesignerWindowS"}

Returns the **Window** object that represents the component's designer.

Remarks

If the component supports a designer but doesn't have an open designer, accessing the **DesignerWindow** property creates the designer, but it isn't visible. To make the window visible, set the **Window** object's **Visible** property to **True**.

CodePanes Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproCodePanesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproCodePanesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproCodePanesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproCodePanesS"}

Returns the collection of active **CodePane** objects. Read-only.

GUID Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproGUIDC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproGUIDX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproGUIDA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproGUIDS"}

Returns a **String** containing the class identifier of an object. Read-only.

HasOpenDesigner Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproHasOpenDesignerC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproHasOpenDesignerX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproHasOpenDesignerA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproHasOpenDesignerS"}

Returns a **Boolean** value indicating whether or not the **VBComponent** object has an open designer. Read-only.

Return Values

The **HasOpenDesigner** property returns these values:

Value	Description
True	The **VBComponent** object has an open **Design** window.
False	The **VBComponent** object doesn't have an open **Design** window.

Height Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproHeightC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproHeightS"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproHeightA"}

Returns or sets a **Single** containing the height of the window in twips. Read/write.

Remarks

Changing the **Height** property setting of a linked window or docked window has no effect as long as the window remains linked or docked.

HelpContextID Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproAddinHelpContextIDC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproAddinHelpContextIDX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproAddinHelpContextIDA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproAddinHelpContextIDS"}

Returns or sets a **String** containing the context ID for a topic in a Microsoft Windows Help file. Read/write.

HelpFile Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproAddinHelpFileC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproAddinHelpFileX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproAddinHelpFileA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproAddinHelpFileS"}

Returns or sets a **String** specifying the Microsoft Windows Help file for a project. Read/write.

IndexedValue Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproIndexedValueC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproIndexedValueX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproIndexedValueA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproIndexedValueS"}

Returns or sets a value for a member of a property that is an indexed list or an array.

Remarks

The value returned or set by the **IndexedValue** property is an expression that evaluates to a type that is accepted by the object. For a property that is an indexed list or array, you must use the **IndexedValue** property instead of the **Value** property. An indexed list is a numeric expression specifying index position.

IndexedValue accepts up to 4 indices. The number of indices accepted by **IndexedValue** is the value returned by the **NumIndices** property.

The **IndexedValue** property is used only if the value of the **NumIndices** property is greater than zero. Values in indexed lists are set or returned with a single index.

IsBroken Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproIsBrokenC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproIsBrokenX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproIsBrokenA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproIsBrokenS"}

Returns a **Boolean** value indicating whether or not the **Reference** object points to a valid reference in the registry. Read-only.

Return Values

The **IsBroken** property returns these values:

Value	Description
True	The **Reference** object no longer points to a valid reference in the registry.
False	The **Reference** object points to a valid reference in the registry.

BuiltIn Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproBuiltInC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproBuiltInX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproBuiltInS"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproBuiltInA"}

Returns a **Boolean** value indicating whether or not the reference is a default reference that can't be removed. Read-only.

Return Values

The **BuiltIn** property returns these values:

Value	Description
True	The reference is a default reference that can't be removed.
False	The reference isn't a default reference; it can be removed.

Saved Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproSavedC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproSavedX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproSavedS"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproSavedA"}

Returns a **Boolean** value indicating whether or not the object was edited since the last time it was saved. Read/write.

Return Values

The **Saved** property returns these values:

Value	Description
True	The object has not been edited since the last time it was saved.
False	The object has been edited since the last time it was saved.

Remarks

The **SaveAs** method sets the **Saved** property to **True**.

Note If you set the **Saved** property to **False** in code, it returns **False**, and the object is marked as if it were edited since the last time it was saved.

Left Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproLeftC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproLeftX":1}
{ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproLeftA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproLeftS"}

Returns or sets a **Single** containing the location of the left edge of the window on the screen in twips. Read/write.

Remarks

The value returned by the **Left** property depends on whether or not the window is linked or docked.

Note Changing the **Left** property setting of a linked or docked window has no effect as long as the window remains linked or docked.

LinkedWindowFrame Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproLinkedWindowFrameC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproLinkedWindowFrameX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproLinkedWindowFrameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproLinkedWindowFrameS"}

Returns the **Window** object representing the frame that contains the window. Read-only.

Remarks

The **LinkedWindowFrame** property enables you to access the object representing the linked window frame, which has properties distinct from the window or windows it contains. If the window isn't linked, the **LinkedWindowFrame** property returns **Nothing**.

FullPath Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproFullPathC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproFullPathX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproFullPathA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproFullPathS"}

Returns a **String** containing the path and file name of the referenced type library. Read-only.

MainWindow Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproMainWindowC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproMainWindowX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproMainWindowA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproMainWindowS"}

Returns a **Window** object representing the main window of the Visual Basic development environment. Read-only.

Remarks

You can use the **Window** object returned by the **MainWindow** property to add or remove docked windows. You can also use the **Window** object returned by the **MainWindow** property to maximize, minimize, hide, or restore the main window of the Visual Basic development environment.

Major Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproMajorC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproMajorX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproMajorA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproMajorS"}

Returns a **Long** containing the major version number of the referenced type library. Read-only.

Remarks

The number returned by the **Major** property corresponds to the major version number stored in the type library to which you have set the reference.

Minor Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproMinorC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproMinorX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproMinorA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproMinorS"}

Returns a **Long** indicating the minor version number of the referenced type library. Read-only.

Remarks

The number returned by the **Minor** property corresponds to the minor version number stored in the type library to which you have set the reference.

Mode Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproModeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproModeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproModeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproModeS"}

Returns a value containing the mode of the specified project. Read-only.

Return Values

The **Mode** property return values are:

Constant	Description
vbext_vm_RunMode	The specified project is in run mode.
vbext_vm_BreakMode	The specified project is in break mode.
vbext_vm_DesignMode	The specified project is in design mode.

Name Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproNameC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproNameX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproNameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproNameS"}

Returns or sets a **String** containing the name used in code to identify an object. For the **VBProject** object and the **VBComponent** object, read/write; for the **Property** object and the **Reference** object, read-only.

Remarks

The following table describes how the **Name** property setting applies to different objects.

Object	Result of Using Name Property Setting
VBProject	Returns or sets the name of the active project.
VBComponent	Returns or sets the name of the component. An error occurs if you try to set the **Name** property to a name already being used or an invalid name.
Property	Returns the name of the property as it appears in the **Property Browser**. This is the value used to index the **Properties** collection. The name can't be set.
Reference	Returns the name of the reference in code. The name can't be set.

The default name for new objects is the type of object plus a unique integer. For example, the first new Form object is Form1, a new Form object is Form1, and the third TextBox control you create on a form is TextBox3.

An object's **Name** property must start with a letter and can be a maximum of 40 characters. It can include numbers and underline (_) characters but can't include punctuation or spaces. Forms and modules can't have the same name as another public object such as **Clipboard**, **Screen**, or **App**. Although the **Name** property setting can be a keyword, property name, or the name of another object, this can create conflicts in your code.

NumIndices Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproNumIndicesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproNumIndicesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproNumIndicesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproNumIndicesS"}

Returns the number of indices on the property returned by the **Property** object.

Remarks

The value of the **NumIndices** property can be an integer from 0 – 4. For most properties, **NumIndices** returns 0. Conventionally indexed properties return 1. Property arrays might return 2.

Object Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproObjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproObjectS"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproObjectA"}

Returns or sets the value of an object returned by a property. Read/write.

Remarks

If a property returns an object, you must use the **Object** property to return or set the value of that object.

Parent Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproParentC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproParentX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproParentA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproParentS"}

Returns the object or collection that contains another object or collection. Read-only.

Remarks

Most objects have either a **Parent** property or a **Collection** property that points to the object's parent object in this object model. The **Collection** property is used if the parent object is a collection.

Use the **Parent** property to access the properties, methods, and controls of an object's parent object.

Protection Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproProtectionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproProtectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproProtectionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproProtectionS"}

Returns a value indicating the state of protection of a project. Read-only.

Return Values

The **Protection** property return values are:

Constant	Description
vbext_pp_locked	The specified project is locked.
vbext_pp_none	The specified project isn't protected.

ReferencesEvents Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproReferencesEventsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproReferencesEventsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproReferencesEventsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproReferencesEventsS"}

Returns the **ReferencesEvents** object. Read-only.

Settings

The setting for the argument you pass to the **ReferencesEvents** property is:

Argument	Description
vbproject	If *vbproject* points to **Nothing**, the object that is returned will supply events for the **References** collections of all **VBProject** objects in the **VBProjects** collection. If *vbproject* points to a valid **VBProject** object, the object that is returned will supply events for only the **References** collection for that project.

Remarks

The **ReferencesEvents** property takes an argument and returns an event source object. The **ReferencesEvents** object is the source for events that are triggered when references are added or removed.

Top Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproTopC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproTopX":1}
{ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproTopA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproTopS"}

Returns or sets a **Single** specifying the location of the top of the window on the screen in twips. Read/write.

Remarks

The value returned by the **Top** property depends on whether or not the window is docked, linked, or in docking view.

Note Changing the **Top** property setting of a linked or docked window has no effect as long as the window remains linked or docked.

TopLine Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproTopLineC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproTopLineX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproTopLineA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproTopLineS"}

Returns a **Long** specifying the line number of the line at the top of the code pane or sets the line showing at the top of the code pane. Read/write.

Remarks

Use the **TopLine** property to return or set the line showing at the top of the code pane. For example, if you want line 25 to be the first line showing in a code pane, set the **TopLine** property to 25.

The **TopLine** property setting must be a positive number. If the **TopLine** property setting is greater than the actual number of lines in the code pane, the setting will be the last line in the code pane.

Type Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproTypeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproTypeX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproTypeS"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproTypeA"}

Returns a numeric or string value containing the type of object. Read-only.

Return Values

The **Type** property settings for the **Window** object are described in the following table:

Constant	Value	Description
vbext_wt_CodeWindow	0	**Code** window
vbext_wt_Designer	1	Designer
vbext_wt_Browser	2	**Object Browser**
vbext_wt_Watch	3	Watch pane
vbext_wt_Locals	4	Locals
vbext_wt_Immediate	5	**Immediate** window
vbext_wt_ProjectWindow	6	**Project** window
vbext_wt_PropertyWindow	7	**Properties** window
vbext_wt_Find	8	**Find** dialog box
vbext_wt_FindReplace	9	**Search and Replace** dialog box
vbext_wt_LinkedWindowFrame	11	Linked window frame
vbext_wt_MainWindow	12	Main window

The **Type** property settings for the **VBComponent** object are described in the following table:

Constant	Description
vbext_ct_ClassModule	Class module
vbext_ct_MSForm	Microsoft Form
vbext_ct_StdModule	Standard module
vbext_ct_Document	Document module

The **Type** property settings for the **Reference** object are described in the following table:

Constant	Description
vbext_rk_TypeLib	Type library
vbext_rk_Project	Project

Value Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproValueC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproValueX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproValueA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproValueS"}

Returns or sets a **Variant** specifying the value of the property. Read/write.

Remarks

Because the **Value** property returns a **Variant**, you can access any property. To access a list, use the **IndexedValue** property.

If the property that the **Property** object represents is read/write, the **Value** property is read/write. If the property is read-only, attempting to set the **Value** property causes an error. If the property is write-only, attempting to return the **Value** property causes an error.

The **Value** property is the default property for the **Property** object.

VBE Property

Returns the root of the **VBE** object. Read-only.

Remarks

All objects have a **VBE** property that points to the root of the **VBE** object.

Version Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproVersionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproVersionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproVersionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproVersionS"}

Returns a **String** containing the version of Visual Basic for Applications that the application is using. Read-only.

Remarks

The **Version** property value is a string beginning with one or two digits, a period, and two digits; the rest of the string is undefined and may contain text or numbers.

Visible Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproVisibleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproVisibleX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproVisibleS"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproVisibleA"}

For the **Window** object, returns or sets a **Boolean** value that specifies the visibility of a window. Read/write. For the **CodePane** object, returns a **Boolean** value that indicates whether or not the code pane is visible in the window. Read-only.

Return Values

The **Visible** property returns the following values:

Value	Description
True	(Default) Object is visible.
False	Object is hidden.

Width Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproWidthC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproWidthS"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproWidthA"}

Returns or sets a **Single** containing the width of the window in twips. Read/write.

Remarks

Changing the **Width** property setting of a linked window or docked window has no effect as long as the window remains linked or docked.

Window Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproWindowC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproWindowX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproWindowA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproWindowS"}

Returns the window in which the code pane is displayed. Read-only.

WindowState Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaproWindowStateC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaproWindowStateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vaproWindowStateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaproWindowStateS"}

Returns or sets a numeric value specifying the visual state of the window. Read/write.

Settings

The **WindowState** property returns or sets the following values:

Constant	Value	Description
vbext_ws_Normal	0	(Default) Normal
vbext_ws_Min	1	Minimized (minimized to an icon)
vbext_ws_Max	2	Maximized (enlarged to maximum size)

ActiveCodePane Property Example

The following example uses the **ActiveCodePane** property and **TopLine** properties to obtain the number of the top line in the active code pane.

```
Debug.Print Application.VBE.ActiveCodePane.TopLine
```

ActiveVBProject Property Example

The following example uses the **ActiveVBProject** property to return the name of the active project.

```
Debug.Print Application.VBE.ActiveVBProject.Name
```

ActiveWindow Property Example

The following example uses the **ActiveWindow** property to return the caption of the active window.

```
Debug.Print Application.VBE.ActiveWindow.Caption
```

BuiltIn Property Example

The following example uses the **BuiltIn** property to return a **Boolean** indicating whether or not a particular reference in the active project is built-in.

```
Debug.Print Application.VBE.ActiveVBProject.References(1).BuiltIn
```

Caption Property Example

The following example uses the **Caption** property to display the caption of the active window.

```
Debug.Print Application.VBE.ActiveWindow.Caption
```

CodeModule Property Example

The following example uses the **CodeModule** and **CountOfLines** properties to return the number of lines in a particular code module.

```
Debug.Print
Application.VBE.ActiveVBProject.VBComponents(6).CodeModule.CountOfLines
```

CodePane Property Example

The following example uses the **CodePane** and **TopLine** properties to display the number of the top line in the code module of the selected **VBComponent** object.

```
Debug.Print Application.VBE.SelectedVBComponent.CodeModule.CodePane.TopLine
```

CodePanes Property Example

The following example uses the **CodePanes** and **TopLine** properties to display the line number of the top line in the specified code pane.

```
Debug.Print Application.VBE.CodePanes(3).TopLine
```

CodePaneView Property Example

The following example uses the **CodePaneView** property to return a value indicating whether the specified code pane is in procedure view or full module view.

```
Debug.Print Application.VBE.CodePanes(3).CodePaneView
```

Collection Property Example

The following example uses the **Collection** and **Count** properties to return the number of objects the active project contains, when viewed as a collection of objects.

```
Debug.Print Application.VBE.ActiveVBProject.Collection.Count
```

CommandBarEvents Property Example

The following example uses code including the **CommandBarEvents** property to support any code to handle a mouse click on a command bar.

```
Private WithEvents ce As CommandBarEvents

Sub Test()
    Dim c As CommandBarControl
    Set c = Application.VBE.CommandBars("Tools").Controls(1)
    Set ce = Application.VBE.Events.CommandBarEvents(c)
End Sub

Private Sub ce_Click(ByVal CommandBarControl As Object, Handled As Boolean,
CancelDefault As Boolean)
    ' Put event-handling code here
End Sub
```

Count Property Example

The following example uses the **Count** property to return the number of **VBComponent** objects in a particular project.

```
Debug.Print Application.VBE.VBProjects(1).VBComponents.Count
```

CountOfDeclarationLines Property Example

The following example uses the **CountOfDeclarationLines** property to return the number of declaration lines in a particular code pane.

```
Debug.Print Application.VBE.CodePanes(2).CodeModule.CountOfDeclarationLines
```

CountOfLines Property Example

The following example uses the **CountOfLines** property to return the total number of lines in a particular code pane.

```
Application.VBE.CodePanes(2).CodeModule.CountOfLines
```

CountOfVisibleLines Property Example

The following example uses the **CountOfVisibleLines** property to return the number of lines visible at one time in a particular code pane, based on the height of the pane.

```
Debug.Print Application.VBE.Codepanes(3).CountOfVisibleLines
```

Description Property Example

The first of the following examples uses the **Description** property to assign a description to a particular project; the example also prints the description to verify that the assignment was successful.

The second example uses the **Description** property to return the descriptive names of the specified **Reference** objects in a particular project.

```
Application.VBE.VBProjects(1).Description = "Hot Sauce"
Debug.Print Application.VBE.VBProjects(1).Description

Debug.Print Application.VBE.VBProjects(1).References(1).Description

Debug.Print Application.VBE.VBProjects(1).References(2).Description
```

Designer Property Example

The following example uses the **Designer** and **Count** properties to return the number of controls on a form. Note that the window containing the form must be selected. The **Designer** object is the form itself.

```
Debug.Print Application.VBE.SelectVBComponent.Designer.Controls.Count
```

DesignerWindow Property Example

The following example uses the **DesignerWindow** and **Visibile** properties to find out whether or not a particular designer is visible. Note that the **VBComponent** object must be a form.

```
Debug.Print
Application.VBE.VBProjects(1).VBComponents(1).DesignerWindow.Visible
```

FullPath Property Example

The following example uses the **FullPath** property to return the full path of the object library for the specified reference.

```
Debug.Print Application.VBE.ActiveVBProject.References(1).FullPath
```

GUID Property Example

The following example uses the **GUID** property to return the globally unique ID number for the specified **Reference** object in the specified project.

```
Debug.Print Application.VBE.VBProjects(1).References(1).GUID
```

HasOpenDesigner Property Example

The following example uses the **HasOpenDesigner** property to return whether or not the specified component, in this case a form, of a particular project has an open designer.

```
Debug.Print Application.VBE.VBProjects(1).VBComponents(1).HasOpenDesigner
```

Height, Width Properties Example

The following example uses the **Height** and **Width** properties to return the height and width of the specified window, in twips. These property settings change after a window is linked or docked because then they refer to the **Window** object to which the original window is linked or docked.

```
Debug.Print Application.VBE.Windows(9).Height
Debug.Print Application.VBE.Windows(9).Width
```

HelpContextID Property Example

The following example uses the **HelpContextID** property to return the context ID for the Help file corresponding to a project.

```
Debug.Print Application.VBE.VBProjects(1).HelpContextID
```

HelpFile Property Example

The following example uses the **HelpFile** property to assign a Help file to a project; the example verifies that the assignment was successful by printing the full path of the Help file.

```
Application.VBE.VBProjects(1).HelpFile = "C:\HelpStuff\veenob3.hlp"
Debug.Print Application.VBE.VBProjects(1).HelpFile
```

IsBroken Property Example

The following example uses the **IsBroken** property to return a value indicating whether or not the specified **Reference** object in a particular project is broken.

```
Debug.Print Application.VBE.vbprojects(1).References(1).IsBroken
```

Left, Top Properties Example

The following example uses the **Left** and **Top** properties to return the coordinates of the upper-left corner of a particular window, in twips. These property settings change after a window is linked or docked because then they refer to the **Window** object to which the original window is linked or docked.

```
Debug.Print Application.VBE.Windows(9).Left
Debug.Print Application.VBE.Windows(9).Top
```

MainWindow Property Example

The following example uses the **MainWindow** property to return the **Window** object representing the main window, and then prints the caption of the main window.

```
Debug.Print Application.VBE.MainWindow.Caption
```

Major Property Example

The following example uses the **Major** property to return the major version number of the specified **Reference** object in a particular project.

```
Debug.Print Application.VBE.VBProjects(1).References(1).Major
```

Minor Property Example

The following example uses the **Minor** property to return the minor version number of the specified **Reference** object in a particular project.

```
Debug.Print Application.VBE.VBProjects(1).References(1).Minor
```

Mode Property Example

The following example uses the **Mode** property to return the mode of the active project. The value returned is a predefined constant representing the project's mode.

```
Debug.Print Application.VBE.ActiveVBProject.Mode
```

Name Property Example

The following example uses the **Name** property to return the name of the specified member of the **VBComponents** collection in a particular project.

```
Debug.Print Application.VBE.VBProjects(1).VBComponents(1).Name
```

NumIndices Property Example

The following example uses the **NumIndices** property to return the number of indexes belonging to the specified property of a particular **VBComponent** object

```
Debug.Print
Application.VBE.VBProjects(1).VBComponents(1).Properties(40).NumIndices
```

Object Property Example

The following example loads the name of an icon into the icon list for the specified object, which must be a form.

```
Set
Application.VBE.ActiveVBProject.VBComponents(1).Properties("Icon").Object =
LoadPicture("Baseball.ico")
```

Parent Property Example

The following example uses the **Parent** property to return the name of an object's parent in the object hierarchy.

```
Debug.Print Application.VBE.ActiveVBProject.VBComponents.Parent.Name
```

Protection Property Example

The following example uses the **Protection** property to return a value indicating whether or not a project is protected. The value returned is a number that corresponds to a predefined constant representing the project's status.

```
Debug.Print Application.VBE.ActiveVBProject.Protection
```

ReferencesEvents Property Example

The following example uses code including the **ReferencesEvents** property to support event-handling code for adding or removing references.

```
Private WithEvents X As ReferencesEvents

Sub Test()
    Set X = Application.VBE.Events.ReferencesEvents
End Sub

Private Sub X_ItemAdded(ByVal Reference As VBIDE.Reference)
    ' Put code to support item addition here
End Sub

Private Sub X_ItemRemoved(ByVal Reference As VBIDE.Reference)
    ' Put code to support item removal here
End Sub
```

Saved Property Example

The following example uses the **Saved** property to return a **Boolean** value indicating whether or not the specified project has been saved in its current state.

```
Debug.Print Application.VBE.VBProjects(1).Saved
```

SelectedVBComponent Property Example

The following example uses the **SelectedVBComponent** property to return the selected component.

```
Debug.Print Application.VBE.SelectedVBComponent.Name
```

TopLine Property Example

The following example uses the **TopLine** property to return the line number of the top line in the specified code pane.

```
Debug.Print Application.VBE.CodePanes(3).TopLine
```

Type Property Example

The following example uses the **Type** property to return a value indicating the type of the specified member of the **VBComponents** collection in a particular project. The value returned is a number that corresponds to a predefined constant for one of the component object types.

```
Debug.Print Application.VBE.VBProjects(1).VBComponents(1).Type
```

Value Property Example

The following example uses the **Value** property to return the value of the specified property of a member of the **VBComponents** collection.

```
Debug.Print Application.VBE.
ActiveVBProject.VBComponents(1).Properties("AcceptLabelsInFormulas").Value
```

VBE Property Example

The following example uses the **VBE** and **Name** properties to return the name of the active project.

```
Debug.Print Application.VBE.ActiveVBProject.Name
```

Version Property Example

The following example uses the **Version** property to return the version number of the host application.

```
Debug.Print Application.VBE.Version
```

Visible Property Example

The following example uses the **Visible** property to return a **Boolean** value indicating whether or not the specified window is visible.

```
Debug.Print Application.VBE.Windows(9).Visible
```

Window Property Example

The following example uses the **Window** and **Caption** properties to return the caption of the specified code pane.

```
Debug.Print Application.VBE.CodePanes(1).Window.Caption
```

WindowState Property Example

The following example uses the **WindowState** property to return the visual state of the specified window. The value returned is a number that corresponds to a predefined constant that specifies the visual state of a window.

```
Debug.Print Application.VBE.Windows(9).WindowState
```

CodeModule Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjCodeModuleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjCodeModuleX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjCodeModuleP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjCodeModuleM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjCodeModuleE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjCodeModuleS"}

Represents the code behind a component, such as a form, class, or document.

Remarks

You use the **CodeModule** object to modify (add, delete, or edit) the code associated with a component.

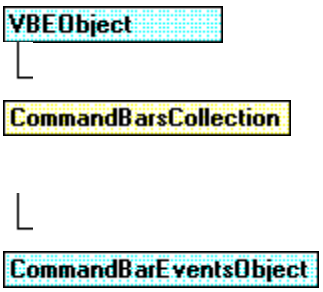
Each component is associated with one **CodeModule** object. However, a **CodeModule** object can be associated with multiple code panes.

The methods associated with the **CodeModule** object enable you to manipulate and return information about the code text on a line-by-line basis. For example, you can use the **AddFromString** method to add text to the module. **AddFromString** places the text just above the first procedure in the module or places the text at the end of the module if there are no procedures.

Use the **Parent** property to return the **VBComponent** object associated with a code module.

CommandBarEvents Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjCommandBarEventsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjCommandBarEventsX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjCommandBarEventsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjCommandBarEventsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjCommandBarEventsE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjCommandBarEventsS"}



Returned by the **CommandBarEvents** property. The **CommandBarEvents** object triggers an event when a control on the command bar is clicked.

Remarks

The **CommandBarEvents** object is returned by the **CommandBarEvents** property of the **Events** object. The object that is returned has one event in its interface, the Click event. You can handle this event using the **WithEvents** object declaration.

CommandBars Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjCommandBarsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjCommandBarsX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjCommandBarsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjCommandBarsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjCommandBarsE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjCommandBarsS"}

Contains all of the command bars in a project, including command bars that support shortcut menus.

Remarks

Use the **CommandBars** collection to enable add-ins to add command bars and controls or to add controls to existing, built-in, command bars.

VBComponent Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjVBComponentC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjVBComponentX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjVBComponentP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjVBComponentM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjVBComponentE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjVBComponentS"}

Represents a component, such as a class module or standard module, contained in a project.

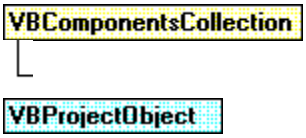
Remarks

Use the **VBComponent** object to access the code module associated with a component or to change a component's property settings.

You can use the **Type** property to find out what type of component the **VBComponent** object refers to. Use the **Collection** property to find out what collection the component is in.

VBComponents Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjVBComponentsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjVBComponentsX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjVBComponentsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjVBComponentsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjVBComponentsE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjVBComponentsS"}



Represents the components contained in a project.

Remarks

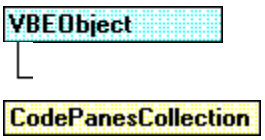
Use the **VBComponents** collection to access, add, or remove components in a project. A component can be a form, module, or class. The **VBComponents** collection is a standard collection that can be used in a **For Each** block.

You can use the **Parent** property to return the project the **VBComponents** collection is in.

In Visual Basic for Applications, you can use **Import** method to add a component to a project from a file.

CodePane Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjCodePaneC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjCodePaneX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjCodePaneP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjCodePaneM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjCodePaneE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjCodePaneS"}



Represents a code pane.

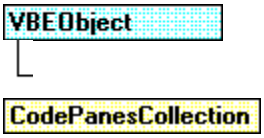
Remarks

Use the **CodePane** object to manipulate the position of visible text or the text selection displayed in the code pane.

You can use the **Show** method to make the code pane you specify visible. Use the **SetSelection** method to set the selection in a code pane and the **GetSelection** method to return the location of the selection in a code pane.

CodePanes Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjCodePanesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjCodePanesX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjCodePanesP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjCodePanesM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjCodePanesE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjCodePanesS"}



Contains the active code panes in the **VBE** object.

Remarks

Use the **CodePanes** collection to access the open code panes in a project.

You can use the **Count** property to return the number of active code panes in a collection.

Events Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjEventsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjEventsX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjEventsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjEventsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjEventsE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjEventsS"}

Supplies properties that enable add-ins to connect to all events in Visual Basic for Applications.

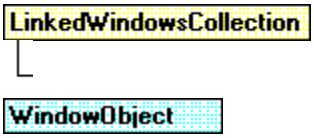
Remarks

The **Events** object provides properties that return event source objects. Use the properties to return event source objects that notify you of changes in the Visual Basic for Applications environment.

The properties of the **Events** object return objects of the same type as the property name. For example, the **CommandBarEvents** property returns the **CommandBarEvents** object.

LinkedWindows Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjLinkedWindowsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjLinkedWindowsX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjLinkedWindowsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjLinkedWindowsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjLinkedWindowsE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjLinkedWindowsS"}



Contains all linked windows in a linked window frame.

Remarks

Use the **LinkedWindows** collection to modify the docked and linked state of windows in the development environment.

The **LinkedWindowFrame** property of the **Window** object returns a **Window** object that has a valid **LinkedWindows** collection.

Linked window frames contain all windows that can be linked or docked. This includes all windows except code windows, designers, the **Object Browser** window, and the **Search and Replace** window.

If all the panes from one linked window frame are moved to another window, the linked window frame with no panes is destroyed. However, if all the panes are removed from the main window, it isn't destroyed.

Use the **Visible** property to check or set the visibility of a window.

You can use the **Add** method to add a window to the collection of currently linked windows. A window that is a pane in one linked window frame can be added to another linked window frame. Use the **Remove** method to remove a window from the collection of currently linked windows; this results in the window being unlinked or undocked.

The **LinkedWindows** collection is used to dock and undock windows from the main window frame.

Property Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjPropertyP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjPropertyM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjPropertyE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjPropertyS"}

Represents the properties of an object that are visible in the **Properties** window for any given component.

Remarks

Use **Value** property of the **Property** object to return or set the value of a property of a component.

At a minimum, all components have a **Name** property. Use the **Value** property of the **Property** object to return or set the value of a property. The **Value** property returns a **Variant** of the appropriate type. If the value returned is an object, the **Value** property returns the **Properties** collection that contains **Property** objects representing the individual properties of the object. You can access each of the **Property** objects by using the **Item** method on the returned **Properties** collection.

If the value returned by the **Property** object is an object, you can use the **Object** property to set the **Property** object to a new object.

Properties Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjPropertiesX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjPropertiesP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjPropertiesM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjPropertiesE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjPropertiesS"}

PropertiesCollection

PropertyObject

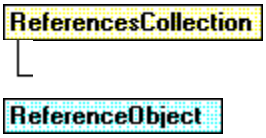
Represents the properties of an object.

Remarks

Use the **Properties** collection to access the properties displayed in the **Properties** window. For every property listed in the **Properties** window, there is an object in the **Properties** collection.

Reference Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjReferenceC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjReferenceX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjReferenceP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjReferenceM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjReferenceE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjReferenceS"}



Represents a reference to a type library or a project.

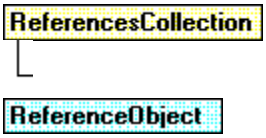
Remarks

Use the **Reference** object to verify whether a reference is still valid.

The **IsBroken** property returns **True** if the reference no longer points to a valid reference. The **BuiltIn** property returns **True** if the reference is a default reference that can't be moved or removed. Use the **Name** property to determine if the reference you want to add or remove is the correct one.

References Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjReferencesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjReferencesX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjReferencesP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjReferencesM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjReferencesE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjReferencesS"}



Represents the set of references in the project.

Remarks

Use the **References** collection to add or remove references. The **References** collection is the same as the set of references selected in the **References** dialog box.

ReferencesEvents Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjReferencesEventsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjReferencesEventsX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjReferencesEventsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjReferencesEventsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjReferencesEventsP"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjReferencesEventsS"}

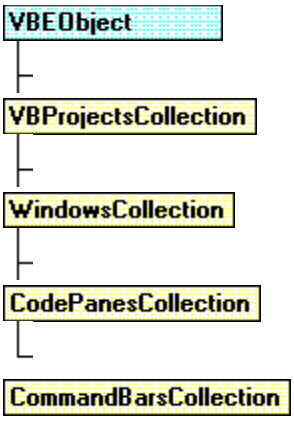
Returned by the **ReferencesEvents** property.

Remarks

The **ReferencesEvents** object is the source of events that occur when a reference is added to or removed from a project. The ItemAdded event is triggered after a reference is added to a project. The ItemRemoved event is triggered after a reference is removed from a project.

VBE Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjVBEC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjVBEX":1}
{ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjVBEP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjVBEM"}
{ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjVBEE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjVBES"}



The root object that contains all other objects and collections represented in Visual Basic for Applications.

Remarks

You can use the following collections to access the objects contained in the **VBE** object:

- Use the **VBProjects** collection to access the collection of projects.
- Use the **Windows** collection to access the collection of windows.
- Use the **CodePanes** collection to access the collection of code panes.
- Use the **CommandBars** collection to access the collection of command bars.

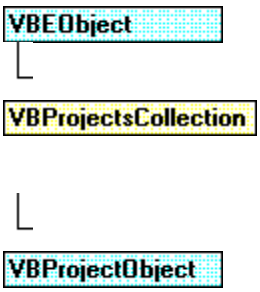
Use the **Events** object to access properties that enable add-ins to connect to all events in Visual Basic for Applications. The properties of the **Events** object return objects of the same type as the property name. For example, the **CommandBarEvents** property returns the **CommandBarEvents** object.

You can use the **SelectedVBComponent** property to return the active component. The active component is the component that is being tracked in the **Project** window. If the selected item in the **Project** window isn't a component, **SelectedVBComponent** returns **Nothing**.

Note All objects in this object model have a **VBE** property that points to the **VBE** object.

VBProject Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjVBProjectC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjVBProjectX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjVBProjectP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjVBProjectM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjVBProjectE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjVBProjects"}



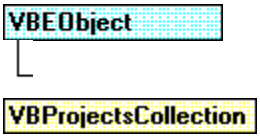
Represents a project.

Remarks

Use the **VBProject** object to set properties for the project, to access the **VBComponents** collection, and to access the **References** collection.

VBProjects Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjVBProjectsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjVBProjectsX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjVBProjectsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjVBProjectsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjVBProjectsE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjVBProjectsS"}



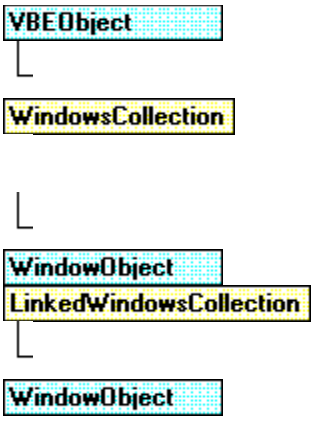
Represents all the projects that are open in the development environment.

Remarks

Use the **VBProjects** collection to access specific projects in an instance of the development environment. **VBProjects** is a standard collection that can be used in a **For Each** block.

Window Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjWindowC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjWindowX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjWindowP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjWindowM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjWindowE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjWindowS"}



Represents a window in the development environment.

Remarks

Use the **Window** object to show, hide, or position windows.

You can use the **Close** method to close a window in the **Windows** collection. The **Close** method affects different types of windows as follows:

Window	Result of using Close method
Code window	Removes the window from the **Windows** collection.
Designer	Removes the window from the **Windows** collection.
Window objects of type linked window frame	Windows become unlinked separate windows.

Note Using the **Close** method with code windows and designers actually closes the window. Setting the **Visible** property to **False** hides the window but doesn't close the window. Using the **Close** method with development environment windows, such as the **Project** window or **Properties** window, is the same as setting the **Visible** property to **False**.

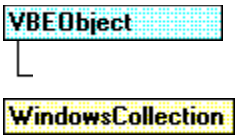
You can use the **SetFocus** method to move the focus to a window.

You can use the **Visible** property to return or set the visibility of a window.

To find out what type of window you are working with, you can use the **Type** property. If you have more than one window of a type, for example, multiple designers, you can use the **Caption** property to determine the window you're working with. You can also find the window you want to work with using the **DesignerWindow** property of the **VBComponent** object or the **Window** property of the **CodePane** object.

Windows Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vaobjWindowsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vaobjWindowsX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vaobjWindowsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vaobjWindowsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vaobjWindowsE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vaobjWindowsS"}



Contains all open or permanent windows.

Remarks

Use the **Windows** collection to access **Window** objects.

The **Windows** collection has a fixed set of windows that are always available in the collection, such as the **Project** window, the **Properties** window, and a set of windows that represent all open code windows and designer windows. Opening a code or designer window adds a new member to the **Windows** collection. Closing a code or designer window removes a member from the **Windows** collection. Closing a permanent development environment window doesn't remove the corresponding object from this collection, but results in the window not being visible.