Appearance Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAuto3DACTIVEXCONTROLSC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAuto3DActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAuto3DActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAuto3DActiveXControlsS"}

Returns or sets the paint style of controls on an **MDIForm** or **Form** object at design time. Read-only at run time.

Syntax

object.**Appearance**

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

Settings

The **Appearance** property settings are:

Setting	Description
0	Flat. Paints controls and forms without visual effects.
1	(Default) 3D. Paints controls with three-dimensional effects.

Remarks

If set to 1 at design time, the **Appearance** property draws controls with three-dimensional effects. If the form's **BorderStyle** property is set to Fixed Double (**vbFixedDouble**, or 3), the caption and border of the form are also painted with three-dimensional effects. Setting the **Appearance** property to 1 also causes the form and its controls to have their **BackColor** property set to the color selected for Button Face in the Color option of the operating system's Control Panel.

Setting the **Appearance** property to 1 for an **MDIForm** object affects only the MDI parent form. To have three-dimensional effects on MDI child forms, you must set each child form's **Appearance** property to 1.

BackColor, ForeColor Properties (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBackColorActiveXControlsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBackColorActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBackColorActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBackColorActiveXControlsS"}

- **BackColor** — returns or sets the background color of an object.
- **ForeColor** — returns or sets the foreground color used to display text and graphics in an object.

Syntax

object.**BackColor** [= *color*]
object.**ForeColor** [= *color*]

The **BackColor** and **ForeColor** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
color	A value or constant that determines the background or foreground colors of an object, as described in Settings.

Settings

Visual Basic uses the Microsoft Windows operating environment red-green-blue (RGB) color scheme. The settings for *color* are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the **RGB** or **QBColor** functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

For all forms and controls, the default settings at design time are:

- **BackColor** — set to the system default color specified by the constant **vbWindowBackground**.
- **ForeColor** — set to the system default color specified by the constant **vbWindowText**.

Remarks

In the **Label**, and **Shape**, controls, the **BackColor** property is ignored if the **BackStyle** property setting is 0 (Transparent).

If you set the **BackColor** property on a **Form** object or a **PictureBox** control, all text and graphics, including the persistent graphics, are erased. Setting the **ForeColor** property doesn't affect graphics or print output already drawn. On all other controls, the screen color changes immediately.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0; the lower 3 bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF). If the high byte isn't 0, Visual Basic uses the system colors, as defined in the user's Control Panel settings and by constants listed in the Visual Basic (VB) object library in the Object Browser.

To display text in the Windows operating environment, both the text and background colors must be

solid. If the text or background colors you've selected aren't displayed, one of the selected colors may be dithered — that is, comprised of up to three different-colored pixels. If you choose a dithered color for either the text or background, the nearest solid color will be substituted.

Note The **Animation** control displays only two types of AVI files, either uncompressed or compressed in RLE8 format. AVI files compressed with RLE8 display only 8-bit colors. The BackColor property for the **Animation** control is "rounded" to the closest 8-bit color in the standard palette.

BorderStyle Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproBorderStyleActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBorderStyleActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBorderStyleActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBorderStyleActiveXControlsS"}

Returns or sets the border style for an object. For the **Form** object and the **TextBox** control, read-only at run time.

Syntax

object.**BorderStyle** = [*value*]

The **BorderStyle** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A value or constant that determines the border style, as described in Settings.

Settings

The **BorderStyle** property settings for a **Form** object are:

Constant	Setting	Description
vbBSNone	0	None (no border or border-related elements).
VbFixedSingle	1	Fixed Single. Can include Control-menu box, title bar, Maximize button, and Minimize button. Resizable only using Maximize and Minimize buttons.
VbSizable	2	(Default) Sizable. Resizable using any of the optional border elements listed for setting 1.
VbFixedDouble	3	Fixed Dialog. Can include Control-menu box and title bar; can't include Maximize or Minimize buttons. Not resizable.
VbFixedToolWindow	4	Fixed ToolWindow. Under Windows 3.x and Windows NT 3.51 and earlier, behaves like Fixed Single. Does not display Maximize or Minimize buttons. Not resizable. Under Windows 95, displays the Close button and displays the title bar text in a reduced font size. The form does not appear in the Windows 95 task bar.
VbSizableToolWindow	5	Sizable ToolWindow. Under Windows 3.x and Windows NT 3.51 and earlier, behaves like Sizable. Does not display Maximize or Minimize buttons. Resizable. Under Windows 95, displays the Close

	button and displays the title bar text in a reduced font size. The form does not appear in the Windows 95 task bar.

The **BorderStyle** property settings for **MS Flex Grid**, **Image**, **Label**, **OLE** container, **PictureBox**, **Frame**, and **TextBox** controls are:

Setting	Description
0	(Default for **Image** and **Label** controls) None.
1	(Default for **MS Flex Grid**, **PictureBox**, **TextBox**, and **OLE** container controls) Fixed Single.

The **BorderStyle** property settings for **Line** and **Shape** controls are:

Constant	Setting	Description
vbTransparent	0	Transparent
vbBSSolid	1	(Default) Solid. The border is centered on the edge of the shape.
vbBSDash	2	Dash
vbBSDot	3	Dot
vbBSDashDot	4	Dash-dot
vbBSDashDotDot	5	Dash-dot-dot
vbBSInsideSolid	6	Inside solid. The outer edge of the border is the outer edge of the shape.

Remarks

For a form, the **BorderStyle** property determines key characteristics that visually identify a form as either a general-purpose window or a dialog box. Setting 3 (Fixed Dialog) is useful for standard dialog boxes. Settings 4 (Fixed ToolWindow) and 5 (Sizable ToolWindow) are useful for creating toolbox-style windows.

MDI child forms set to 2 (Sizable) are displayed within the MDI form in a default size defined by the Windows operating environment at run time. For any other setting, the form is displayed in the size specified at design time.

Changing the setting of the **BorderStyle** property of a **Form** object may change the settings of the **MinButton**, **MaxButton**, and **ShowInTaskbar** properties. When **BorderStyle** is set to 1 (Fixed Single) or 2 (Sizable), the **MinButton**, **MaxButton**, and **ShowInTaskbar** properties are automatically set to **True**. When **BorderStyle** is set to 0 (None), 3 (Fixed Dialog), 4 (Fixed ToolWindow), or 5 (Sizable ToolWindow), the **MinButton**, **MaxButton**, and **ShowInTaskbar** properties are automatically set to **False**.

Note If a form with a menu is set to 3 (Fixed Dialog), it is displayed with a setting 1 (Fixed Single) border instead.

At run time, a form is either modal or modeless, which you specify using the **Show** method.

BorderStyle Constants (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbidxBorderStyleConstantsACTIVEXCONTROLSC;vbproBooksOnlineJumpTopic"}

Constant	Value	Description
ccNone	0	(Default) No border or border-related elements.
ccFixedSingle	1	(Default for **ListView** control) Fixed single. There is a single line border around the control.

Note The cc prefix refers to the custom controls. The prefixes for the constants change with the specific control or group of controls. However, the description remains the same unless indicated.

Caption Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCaptionActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCaptionActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCaptionActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCaptionActiveXControlsS"}

- Form — determines the text displayed in the **Form** or **MDIForm** object's title bar . When the form is minimized, this text is displayed below the form's icon.
- Control — determines the text displayed in or next to a control.
- **MenuLine** object — determines the text displayed for a **Menu** control or an object in the **MenuItems** collection.

For a **Menu** control, **Caption** is normally read/write at run time. But **Caption** is read-only for menus that are exposed or supplied by Visual Basic to add-ins, such as the **MenuLine** object.

Syntax

object.**Caption** [= *string*]

The **Caption** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form associated with the active form moduleis assumed to be *object.*
string	A string expression that evaluates to the text displayed as the caption.

Remarks

When you create a new object, its default caption is the default **Name** property setting. This default caption includes the object name and an integer, such as Command1 or Form1. For a more descriptive label, set the **Caption** property.

You can use the **Caption** property to assign an access key to a control. In the caption, include an ampersand (&) immediately preceding the character you want to designate as an access key. The character is underlined. Press the ALT key plus the underlined character to move the focus to that control. To include an ampersand in a caption without creating an access key, include two ampersands (&&). A single ampersand is displayed in the caption and no characters are underlined.

A **Label** control's caption size is unlimited. For forms and all other controls that have captions, the limit is 255 characters.

To display the caption for a form, set the **BorderStyle** property to either Fixed Single (1 or **vbFixedSingle**), Sizable (2 or **vbSizable**), or Fixed Double (3 or **vbFixedDouble**). A caption too long for the form's title bar is clipped. When an MDI child form is maximized within an **MDIForm** object, the child form's caption is included in the parent form's caption.

Tip For a label, set the **AutoSize** property to **True** to automatically resize the control to fit its caption.

Change Event (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtChangeACTIVEXCONTROLSC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtChangeActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtChangeActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtChangeActiveXControlsS"}

Indicates the contents of a control have changed. How and when this event occurs varies with the control:

- **ComboBox** — changes the text in the text box portion of the control. Occurs only if the **Style** property is set to 0 (Dropdown Combo) or 1 (Simple Combo) and the user changes the text or you change the **Text** property setting through code.
- **DirListBox** — changes the selected directory. Occurs when the user double-clicks a new directory or when you change the **Path** property setting through code.
- **DriveListBox** — changes the selected drive. Occurs when the user selects a new drive or when you change the **Drive** property setting through code.
- **HScrollBar** and **VScrollBar** (horizontal and vertical scroll bars) — move the scroll box portion of the scroll bar. Occurs when the user scrolls or when you change the **Value** property setting through code.
- **Label** — changes the contents of the **Label**. Occurs when a DDE link updates data or when you change the **Caption** property setting through code.
- **PictureBox** — changes the contents of the **PictureBox**. Occurs when a DDE link updates data or when you change the **Picture** property setting through code.
- **TextBox** — changes the contents of the text box. Occurs when a DDE link updates data, when a user changes the text, or when you change the **Text** property setting through code.

Syntax

Private Sub *object*_**Change(**[*index* **As Integer**]**)**

The Change event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array

Remarks

The Change event procedure can synchronize or coordinate data display among controls. For example, you can use a scroll bar's Change event procedure to update the scroll bar's **Value** property setting in a **TextBox** control. Or you can use a Change event procedure to display data and formulas in a work area and results in another area.

Change event procedures are also useful for updating properties in file-system controls (**DirListBox**, **DriveListBox**, and **FileListBox**). For example, you can update the **Path** property setting for a **DirListBox** control to reflect a change in a **DriveListBox** control's **Drive** property setting.

Note A Change event procedure can sometimes cause a cascading event. This occurs when the control's Change event alters the control's contents, for example, by setting a property in code that determines the control's value, such as the **Text** property setting for a **TextBox** control. To prevent a cascading event:

- If possible, avoid writing a Change event procedure for a control that alters that control's contents. If you do write such a procedure, be sure to set a flag that prevents further changes while the current change is in progress.
- Avoid creating two or more controls whose Change event procedures affect each other, for

example, two **TextBox** controls that update each other during their Change events.

- Avoid using a **MsgBox** function or statement in this event for **HScrollBar** and **VScrollBar** controls.

Clear Method (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthClearObjectACTIVEXCONTROLSC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthClearMethodX;vbmthClearObjectsActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthClearObjectsActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthClearObjectsActiveXControlsS"}

Removes all objects in a collection.

Syntax

*object.***Clear**

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

Remarks

To remove only one object from a collection, use the **Remove** method.

Clear Method (ActiveX Controls) Example

This example adds six **Panel** objects to a **StatusBar** control, creating a total of seven **Panel** objects. A click on the form clears all **Panel** objects when their number reaches seven. If the number of **Panel** objects is less than seven, each click on the form will add a new **Panel** object to the control until the number seven is once again reached. To try the example, place a **StatusBar** control on a form and paste the code into the Declarations section. Run the example and click on the form to clear all **Panel** objects and subsequently add **Panel** objects.

```
Private Sub Form_Load()
   Dim pnlX As Panel ' Declare object variable for Panel objects.
   Dim I As Integer

   ' Add 6 Panel objects to the single default Panel object,
   ' making 7 Panel objects.
   For I = 1 to 6
      Set pnlX = StatusBar1.Panels.Add
   Next I
End Sub

Private Sub Form_Click()
   ' If the Count of the collection is 7, then clear the collection.
   ' Otherwise, add one Panel and use the collection's Count property
   ' to set its Style.
   If StatusBar1.Panels.Count = 7 Then
      StatusBar1.Panels.Clear
   Else
      Dim pnlX As Panel
      Set pnlX = StatusBar1.Panels.Add( , , "simple", 0)
      ' The Style property is enumerated from 0 to 6. Use the Panels
      ' Count property -1 to set the Style property for the new Panel.
      ' Display all panels regardless of form width.
      pnlX.minwidth = TextWidth("simple")
      pnlX.AutoSize = sbrSpring
      pnlX.Style = Statusbar1.Panels.Count - 1
   End If
End Sub
```

Click Event (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtClickACTIVEXCONTROLSC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtClickActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtClickActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtClickActiveXControlsS"}

Occurs when the user presses and then releases a mouse button over an object. It can also occur when the value of a control is changed.

For a **Form** object, this event occurs when the user clicks either a blank area or a disabled control. For a control, this event occurs when the user:

- Clicks a control with the left or right mouse button. With a **CheckBox**, **CommandButton**, **Listbox**, or **OptionButton** control, the Click event occurs only when the user clicks the left mouse button.
- Selects an item in a **ComboBox** or **ListBox** control, either by pressing the arrow keys or by clicking the mouse button.
- Presses the SPACEBAR when a **CommandButton**, **OptionButton**, or **CheckBox** control has the focus.
- Presses ENTER when a form has a **CommandButton** control with its **Default** property set to **True**.
- Presses ESC when a form has a Cancel button — a **CommandButton** control with its **Cancel** property set to **True**.
- Presses an access key for a control. For example, if the caption of a **CommandButton** control is "&Go", pressing ALT+G triggers the event.

You can also trigger the Click event in code by:

- Setting a **CommandButton** control's **Value** property to **True**.
- Setting an **OptionButton** control's **Value** property to **True**.
- Changing a **CheckBox** control's **Value** property setting.

Syntax

Private Sub Form_Click( )
Private Sub *object*_**Click(**[*index* **As Integer**]**)**

The Click event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

Typically, you attach a Click event procedure to a **CommandButton** control, **Menu** object, or **PictureBox** control to carry out commands and command-like actions. For the other applicable controls, use this event to trigger actions in response to a change in the control.

You can use a control's **Value** property to test the state of the control from code. Clicking a control generates MouseDown and MouseUp events in addition to the Click event. The order in which these three events occur varies from control to control. For example, for **ListBox** and **CommandButton** controls, the events occur in this order: MouseDown, Click, MouseUp. But for **FileListBox**, **Label**, or **PictureBox** controls, the events occur in this order: MouseDown, MouseUp, and Click. When you're attaching event procedures for these related events, be sure that their actions don't conflict. If the order of events is important in your application, test the control to determine the event order.

Note To distinguish between the left, right, and middle mouse buttons, use the MouseDown and MouseUp events.

If there is code in the Click event, the DlbClick event will never trigger, because the Click event is the first event to trigger between the two. As a result, the mouse click is intercepted by the Click event, so the DblClick event doesn't occur.

Clipboard Object Constants (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbidxClipboardConstantsACTIVEXCONTROLSC;vbproBooksOnlineJumpTopic"}

Constant	Value	Description
vbCFLink	&HBF00	DDE conversation information
vbCFRTF	&HBF01	Rich Text Format (.rtf file)
vbCFText	1	Text (.txt file)
vbCFBitmap	2	Bitmap (.bmp file)
vbCFMetafile	3	Metafile (.wmf file)
vbCFDIB	8	Device-independent bitmap
vbCFPalette	9	Color palette

Count Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCountActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCountActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCountActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCountActiveXControlsS"}

Returns the number of objects in a collection.

Syntax

object.**Count**

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

Remarks

You can use this property with a **For...Next** statement to carry out an operation on the forms or controls in a collection. For example, the following code moves all controls on a form 0.5 inches to the right (**ScaleMode** property setting is 1 or **vbTwips**):

```
For I = 0 To Form1.Controls.Count - 1
   Form1.Controls(I).Left = Form1.Controls(I).Left + 720
Next I
```

You can also use this kind of structure to quickly enable or disable all controls on a form.

When used with the **If TypeOf** statement, you can cycle through all controls and change, for example, the **Enabled** property setting of only the text boxes or the **BackColor** property setting of only the option buttons.

Enabled Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproEnabledActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproEnabledActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproEnabledActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproEnabledActiveXControlsS"}

Returns or sets a value that determines whether a form or control can respond to user-generated events.

Syntax

object.**Enabled** [= *boolean*]

The **Enabled** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form associated with the active form module is assumed to be *object.*
boolean	A Boolean expression that specifies whether *object* can respond to user-generated events.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Allows *object* to respond to events.
False	Prevents *object* from responding to events.

Remarks

The **Enabled** property allows forms and controls to be enabled or disabled at run time. For example, you can disable objects that don't apply to the current state of the application. You can also disable a control used purely for display purposes, such as a text box that provides read-only information.

Disabling a **Timer** control by setting **Enabled** to **False** cancels the countdown set up by the control's **Interval** property.

For a **Menu** control, **Enabled** is normally read/write at run time. But **Enabled** is read-only for menu items that are exposed or supplied by Visual Basic to add-ins, such as the Add-In Manager command on the Add-Ins menu.

DataObject Object (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjDataObjectACTIVEXCONTROLSC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjDataObjectActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjDataObjectActiveXControlsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjDataObjectActiveXControlsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjDataObjectActiveXControlsE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjDataObjectActiveXControlsS"}

The **DataObject** object is a container for data being transferred from an component source to an component target. The data is stored in the format defined by the method using the **DataObject** object.

Syntax

DataObject

Remarks

The **DataObject**, which mirrors the **IDataObject** interface, allows OLE drag and drop and clipboard operations to be implemented.

Most components support manual OLE drag and drop events, and some support automatic OLE drag and drop events.

DataObjectFiles Collection (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolDataObjectFilesACTIVEXCONTROLSC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolDataObjectFilesActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vamthItem;vbcolDataObjectFilesActiveXControlsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vamthAdd;vbcolDataObjectFilesActiveXControlsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbcolDataObjectFilesActiveXControlsE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcolDataObjectFilesActiveXControlsS"}

A collection whose elements represent a list of all filenames used by a **DataObject** object (such as the names of files that a user drags to or from the Windows File Explorer.)

Syntax

object.**DataObjectFiles(***index***)**

The **DataObjectFiles** collection syntax has these parts:

Part	Description
object	An object expression that evaluates to a **DataObject** object.
index	An integer with a range from 0 to `DataObjectFiles.Count - 1`.

Remarks

Note This collection is used by the **Files** property only when the data in the **DataObject** object is in the **vbCFFiles** format.

The **DataObjectFiles** collection is used by the **Files** property to store filenames in a **DataObject** object. It includes the **Remove**, **Add**, and **Clear** methods which allow you to manipulate its contents.

DataSource Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDataSourceActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataSourceActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataSourceActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataSourceActiveXControlsS"}

Sets a value that specifies the **Data** control through which the current control is bound to a database. Not available at run time.

Remarks

To bind a control to a field in a database at run time, you must specify a **Data** control in the **DataSource** property at design time using the Properties window.

To complete the connection with a field in the **Recordset** managed by the **Data** control, you must also provide the name of a **Field** object in the **DataField** property. Unlike the **DataField** property, the **DataSource** property setting isn't available at run time.

Data Type

String

DblClick Event (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"formDblClickSee;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtDblClickActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"formDblClickActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtDblClickActiveXControlsS"}

Occurs when the user presses and releases a mouse button and then presses and releases it again over an object.

For a form, the DblClick event occurs when the user double-clicks a disabled control or a blank area of a form. For a control, it occurs when the user:

- Double-clicks a control with the left mouse button.
- Double-clicks an item in a **ComboBox** control whose **Style** property is set to 1 (Simple) or in a **FileListBox**, **ListBox**, **DBCombo**, or **DBList** control.

Syntax

Private Sub Form_DblClick ( )
Private Sub *object*_**DblClick (***index* **As Integer)**

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Identifies the control if it's in a control array.

Remarks

The argument *Index* uniquely identifies a control if it's in a control array. You can use a DblClick event procedure for an implied action, such as double-clicking an icon to open a window or document. You can also use this type of procedure to carry out multiple steps with a single action, such as double-clicking to select an item in a list box and to close the dialog box.

To produce such shortcut effects in Visual Basic, you can use a DblClick event procedure for a list box or file list box in tandem with a default button — a **CommandButton** control with its **Default** property set to **True**. As part of the DblClick event procedure for the list box, you simply call the default button's Click event.

For those objects that receive Mouse events, the events occur in this order: MouseDown, MouseUp, Click, DblClick, and MouseUp.

If DblClick doesn't occur within the system's double-click time limit, the object recognizes another Click event. The double-click time limit may vary because the user can set the double-click speed in the Control Panel. When you're attaching procedures for these related events, be sure that their actions don't conflict. Controls that don't receive DblClick events may receive two clicks instead of a DblClick.

Note To distinguish between the left, right, and middle mouse buttons, use the MouseDown and MouseUp events.

If there is code in the Click event, the DlbClick event will never trigger.

DblClick Event (ActiveX Controls) Example

This example displays a selected list item in a **TextBox** control when either a **CommandButton** control is clicked or a list item is double-clicked. To try this example, paste the code into the Declarations section of a **Form** object that contains a **ListBox** control, a **TextBox** control, and a **CommandButton** control. Then run the example and click the **CommandButton** control or double-click an item in the **ListBox** control.

```
Private Sub Form_Load ()
   List1.AddItem "John" ' Add list box entries.
   List1.AddItem "Paul"
   List1.AddItem "George"
   List1.AddItem "Ringo"
End Sub

Private Sub List1_DblClick ()
   Command1.Value = True   ' Trigger Click event.
End Sub

Private Sub Command1_Click ()
   Text1.Text = List1.Text ' Display selection.
End Sub
```

Drag-and-Drop Constants (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbidxDragandDropConstantsACTIVEXCONTROLSC;vbproBooksOnlineJumpTopic"}

DragOver Event

Constant	Value	Description
vbEnter	0	Source control dragged into target
vbLeave	1	Source control dragged out of target
vbOver	2	Source control dragged from one position in target to another

Drag Method (Controls)

Constant	Value	Description
vbCancel	0	Cancel drag operation
vbBeginDrag	1	Begin dragging control
vbEndDrag	2	Drop control

DragMode Property

Constant	Value	Description
vbManual	0	Manual
vbAutomatic	1	Automatic

FetchVerbs Method (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthFetchVerbsACTIVEXCONTROLSC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthFetchVerbsActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthFetchVerbsActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthFetchVerbsActiveXControlsS"}

Updates the list of verbs an object supports.

Syntax

object.**FetchVerbs**

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

Remarks

You can read the updated list of verbs using the **ObjectVerbs** property.

Files Method (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthFilesMethodACTIVEXCONTROLSC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthFilesMethodActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthFilesMethodActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthFilesMethodActiveXControlsS"}

Returns a collection of filenames used by the vbCFFiles format (a **DataObjectFiles** collection) which in turn contains a list of all filenames used by a **DataObject** object; for example, the names of files that a user drags to or from the Windows File Explorer.

Syntax

object.**Files(***index***)**

The **Files** collection syntax has these parts:

Part	Description
object	An object expression that evaluates to a **DataObject** object.
index	An integer which is an index to an array of filenames.

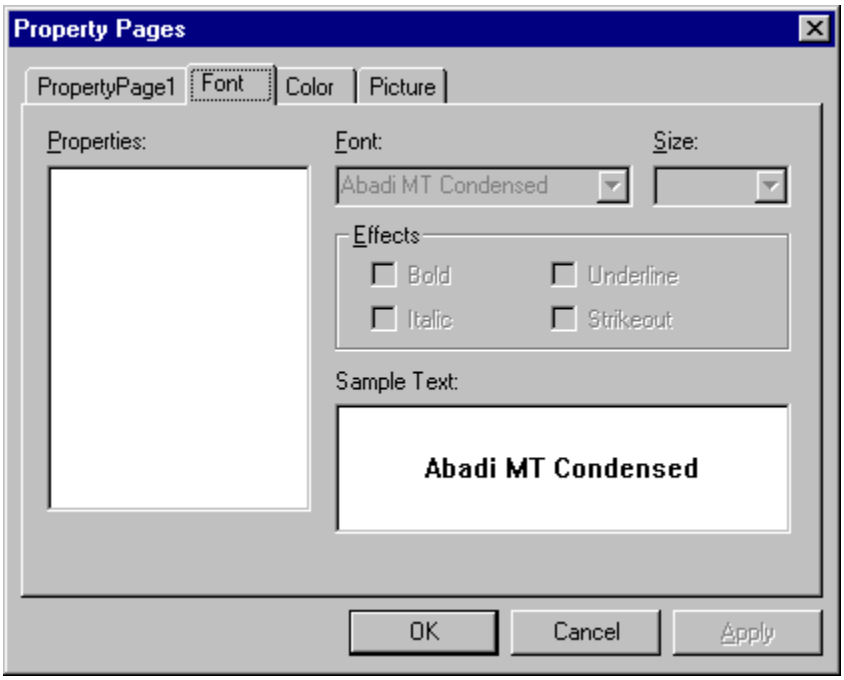
Remarks

The **Files** collection is filled with filenames only when the **DataObject** object contains data of type **vbCFFiles**. The **DataObject** object can contain several different types of data. You can iterate through the collection to retrieve the list of file names.

The **Files** collection can be filled to allow Visual Basic applications to act as a drag source for a list of files.

Property Pages Dialog Box (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbdlgPropertyPagesDialogACTIVEXCONTROLSC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbdlgPropertyPagesDialogActiveXControlsS"}



Allows you to change a control's properties at design time.

Dialog Box Options

Tabs Visual Basic creates a tabbed dialog box that acts like form by writing code to handle updating property values when a user changes values in the control.

You can add Property Pages to your project using the Add Property Pages command on the Project menu.

OK Adds the Property Pages and closes the Property Pages dialog box.

Apply Adds the Property Page without closing the dialog box.

No fonts exist (Error 24574) (Common Dialog Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgNoFontsExistError24574CommonDialogControlC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgNoFontsExistError24574CommonDialogControlS"}

Before displaying the Choose Font dialog box, you must set one of the following flags:

- **ScreenFonts**
- **PrinterFonts**
- **Both**

Font Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontActiveXControlsS"}

Returns a **Font** object.

Syntax

object.**Font**

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

Remarks

Use the **Font** property of an object to identify a specific **Font** object whose properties you want to use. For example, the following code changes the **Bold** property setting of a **Font** object identified by the **Font** property of a **TextBox** object:

```
txtFirstName.Font.Bold = True
```

FontName Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontNameActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontNameActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontNameActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontNameActiveXControlsS"}

Returns or sets the font used to display text in a control or in a run-time drawing or printing operation.

Note The **FontName** property is included for use with the **CommonDialog** control and for compatibility with earlier versions of Visual Basic. For additional functionality, use the new **Font** object properties (not available for the **CommonDialog** control).

Syntax

object.**FontName** [= *font*]

The **FontName** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
font	A string expression specifying the font name to use.

Remarks

The default for this property is determined by the system. Fonts available with Visual Basic vary depending on your system configuration, display devices, and printing devices. Font-related properties can be set only to values for which fonts exist.

In general, you should change **FontName** before setting size and style attributes with the **FontSize**, **FontBold**, **FontItalic**, **FontStrikethru**, and **FontUnderline** properties.

Note At run time, you can get information on fonts available to the system through the **FontCount** and **Fonts** properties.

FontSize Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontSizeActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontSizeActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontSizeActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontSizeActiveXControlsS"}

Returns or sets the size of the font to be used for text displayed in a control or in a run-time drawing or printing operation.

Note The **FontSize** property is included for use with the **CommonDialog** control and for compatibility with earlier versions of Visual Basic. For additional functionality, use the new **Font** object properties (not available for the **CommonDialog** control).

Syntax

object.**FontSize** [= *points*]

The **FontSize** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
points	A numeric expression specifying the font size to use, in points.

Remarks

Use this property to format text in the font size you want. The default is determined by the system. To change the default, specify the size of the font in points.

The maximum value for **FontSize** is 2160 points.

Note Fonts available with Visual Basic vary depending on your system configuration, display devices, and printing devices. Font-related properties can be set only to values for which fonts exist.

In general, you should change the **FontName** property before you set size and style attributes with the **FontSize**, **FontBold**, **FontItalic**, **FontStrikethru**, and **FontUnderline** properties. However, when you set TrueType fonts to smaller than 8 points, you should set the point size with the **FontSize** property, then set the **FontName** property, and then set the size again with the **FontSize** property. The Microsoft Windows operating environment uses a different font for TrueType fonts that are smaller than 8 points.

GetData Method (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetDataActiveXControlsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetDataActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetDataActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetDataActiveXControlsS"}

Returns a graphic from the **Clipboard** object. Doesn't support named arguments.

Syntax

object.**GetData (***format***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	Optional. A constant or value that specifies the **Clipboard** graphics format, as described in Settings. Parentheses must enclose the constant or value. If *format* is 0 or omitted, **GetData** automatically uses the appropriate format.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	metafile (.wmf files)
vbCFDIB	8	Device-independent bitmap (DIB)
vbCFPalette	9	Color palette

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

If no graphic on the **Clipboard** object matches the expected format, nothing is returned. If only a color palette is present on the **Clipboard** object, a minimum size (1 x 1) DIB is created.

GetFormat Method (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetFormatActiveXControlsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetFormatActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetFormatActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetFormatActiveXControlsS;vbmthGetFormatS"}

Returns an integer indicating whether an item on the **Clipboard** object matches a specified format. Doesn't support named argument.

Syntax

object.**GetFormat (***format***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	Required. A value or constant that specifies the **Clipboard** object format, as described in Settings. Parentheses must enclose the constant or value.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFLink	&HBF00	DDE conversation information
vbCFText	1	Text
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	Metafile (.wmf files)
vbCFDIB	8	Device-independent bitmap (DIB)
vbCFPalette	9	Color palette

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

The **GetFormat** method returns **True** if an item on the **Clipboard** object matches the specified format. Otherwise, it returns **False**.

For **vbCFDIB** and **vbCFBitmap** formats, whatever color palette is on the **Clipboard** is used when the graphic is displayed.

HideSelection Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHideSelectionActiveXC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHideSelectionActiveXX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHideSelectionActiveXA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHideSelectionActiveXS"}

Returns a value that determines whether selected text appears highlighted when a control loses the focus.

Syntax

object.**HideSelection**

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

Return Values

The **HideSelection** property return values are:

Value	Description
True	(Default) Selected text doesn't appear highlighted when the control loses the focus.
False	Selected text appears highlighted when the control loses the focus.

Remarks

You can use this property to indicate which text is highlighted while another form or a dialog box has the focus — for example, in a spell-checking routine.

hWnd Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbprohWndActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbprohWndActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbprohWndActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbprohWndActiveXControlsS"}

Returns a handle to a form or control.

Note This property is not supported for the **OLE** container control.

Syntax

object.**hWnd**

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

Remarks

The Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or **hWnd**. The **hWnd** property is used with Windows API calls. Many Windows operating environment functions require the **hWnd** of the active window as an argument.

Note Because the value of this property can change while a program is running, never store the **hWnd** value in a variable.

Item Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthItemMethodActiveXControlsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthItemMethodActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthItemMethodActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthItemMethodActiveXControlsS"}

Returns a specific member of a **Collection** object either by position or by key.

Syntax

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

The **Item** property syntax has the following object qualifier and part:

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 referred to was added to the collection.

Remarks

If the value provided as index doesn’t match any existing member of the collection, an error occurs.

Item is the default property for a collection. Therefore, the following lines of code are equivalent:

```
Print MyCollection(1)
Print MyCollection.Item(1)
```

Key Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproKeyActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproKeyActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproKeyActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproKeyActiveXControlsS"}

Returns or sets a string that uniquely identifies a member in a collection.

Syntax

object.**Key** [= *string*]

The **Key** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A unique string identifying a member in a collection.

Remarks

If the string is not unique, an error will occur.

You can set the **Key** property when you use the **Add** method to add an object to a collection.

The value of the **Index** property of an object can change when objects in the collection are reordered, such as when you set the **Sorted** property to **True**. If you expect the **Index** property to change dynamically, refer to objects in a collection using the **Key** property.

In addition, you can use the **Key** property to make your Visual Basic project "self-documenting" by assigning meaningful names to the objects in a collection.

KeyDown, KeyUp Events (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtKeyDownActiveXControlsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtKeyDownActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtKeyDownActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtKeyDownActiveXControlsS"}

Occur when the user presses (KeyDown) or releases (KeyUp) a key while an object has the focus. (To interpret ANSI characters, use the KeyPress event.)

Syntax

Private Sub Form_KeyDown(*keycode* **As Integer**, *shift* **As Integer)**
Private Sub *object*_**KeyDown(**[*index* **As Integer**,]*keycode* **As Integer**, *shift* **As Integer)**
Private Sub Form_KeyUp(*keycode* **As Integer**, *shift* **As Integer)**
Private Sub *object*_**KeyUp(**[*index* **As Integer**,]*keycode* **As Integer**, *shift* **As Integer)**

The KeyDown and KeyUp event syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
keycode	A key code, such as **vbKeyF1** (the F1 key) or **vbKeyHome** (the HOME key). To specify key codes, use the constants in the Visual Basic (VB) object library in the Object Browser.
shift	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The *shift* argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of *shift* is 6.

Remarks

For both events, the object with the focus receives all keystrokes. A form can have the focus only if it has no visible and enabled controls. Although the KeyDown and KeyUp events can apply to most keys, they're most often used for:

- Extended character keys such as function keys.
- Navigation keys.
- Combinations of keys with standard keyboard modifiers.
- Distinguishing between the numeric keypad and regular number keys.

Use KeyDown and KeyUp event procedures if you need to respond to both the pressing and releasing of a key.

KeyDown and KeyUp aren't invoked for:

- The ENTER key if the form has a **CommandButton** control with the **Default** property set to **True**.
- The ESC key if the form has a **CommandButton** control with the **Cancel** property set to **True**.
- The TAB key.

KeyDown and KeyUp interpret the uppercase and lowercase of each character by means of two arguments: *keycode*, which indicates the physical key (thus returning A and a as the same key) and *shift*, which indicates the state of *shift*+*key* and therefore returns either A or a.

If you need to test for the *shift* argument, you can use the *shift* constants which define the bits within the argument. The constants have the following values:

Constant	Value	Description
vbShiftMask	1	SHIFT key bit mask.
VbCtrlMask	2	CTRL key bit mask.
VbAltMask	4	ALT key bit mask.

The constants act as bit masks that you can use to test for any combination of keys.

You test for a condition by first assigning each result to a temporary integer variable and then comparing *shift* to a bit mask. Use the **And** operator with the *shift* argument to test whether the condition is greater than 0, indicating that the modifier was pressed, as in this example:

```
ShiftDown = (Shift And vbShiftMask) > 0
```

In a procedure, you can test for any combination of conditions, as in this example:

```
If ShiftDown And CtrlDown Then
```

Note If the **KeyPreview** property is set to **True**, a form receives these events before controls on the form receive the events. Use the **KeyPreview** property to create global keyboard-handling routines.

KeyPress Event (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtKeyPressActiveXControlsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtKeyPressActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtKeyPressActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtKeyPressActiveXControlsS"}

Occurs when the user presses and releases an ANSI key.

Syntax

Private Sub Form_KeyPress(*keyascii* **As Integer)**
Private Sub *object*_**KeyPress(**[*index* **As Integer**,]*keyascii* **As Integer)**

The KeyPress event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
keyascii	An integer that returns a standard numeric ANSI keycode. *Keyascii* is passed by reference; changing it sends a different character to the object. Changing *keyascii* to 0 cancels the keystroke so the object receives no character.

Remarks

The object with the focus receives the event. A form can receive the event only if it has no visible and enabled controls or if the **KeyPreview** property is set to **True**. A KeyPress event can involve any printable keyboard character, the CTRL key combined with a character from the standard alphabet or one of a few special characters, and the ENTER or BACKSPACE key. A KeyPress event procedure is useful for intercepting keystrokes entered in a **TextBox** or **ComboBox** control. It enables you to immediately test keystrokes for validity or to format characters as they're typed. Changing the value of the *keyascii* argument changes the character displayed.

You can convert the *keyascii* argument into a character by using the expression:

```
Chr(KeyAscii)
```

You can then perform string operations and translate the character back to an ANSI number that the control can interpret by using the expression:

```
KeyAscii = Asc(char)
```

Use KeyDown and KeyUp event procedures to handle any keystroke not recognized by KeyPress, such as function keys, editing keys, navigation keys, and any combinations of these with keyboard modifiers. Unlike the KeyDown and KeyUp events, KeyPress doesn't indicate the physical state of the keyboard; instead, it passes a character.

KeyPress interprets the uppercase and lowercase of each character as separate key codes and, therefore, as two separate characters. KeyDown and KeyUp interpret the uppercase and lowercase of each character by means of two arguments: *keycode*, which indicates the physical key (thus returning A and a as the same key), and *shift*, which indicates the state of *shift+key* and therefore returns either A or a.

If the **KeyPreview** property is set to **True**, a form receives the event before controls on the form receive the event. Use the **KeyPreview** property to create global keyboard-handling routines.

Note The ANSI number for the keyboard combination of CTRL+@ is 0. Because Visual Basic recognizes a *keyascii* value of 0 as a zero-length string (""), avoid using CTRL+@ in your applications.

MouseDown, MouseUp Events (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtMouseDownActiveXControlsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtMouseDownActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtMouseDownActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtMouseDownActiveXControlsS"}

Occur when the user presses (MouseDown) or releases (MouseUp) a mouse button.

Syntax

Private Sub Form_MouseDown(*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub MDIForm_MouseDown(*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub *object***_MouseDown([***index* **As Integer**,]*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub Form_MouseUp(*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub MDIForm_MouseUp(*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub *object* **_MouseUp([***index* **As Integer**,]*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**

The MouseDown and MouseUp event syntaxes have these parts:

Part	Description
object	Returns an object expression that evaluates to an object in the Applies To list.
index	Returns an integer that uniquely identifies a control if it's in a control array.
button	Returns an integer that identifies the button that was pressed (MouseDown) or released (MouseUp) to cause the event. The *button* argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.
shift	Returns an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the *button* argument is pressed or released. A bit is set if the key is down. The *shift* argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The *shift* argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of *shift* would be 6.
x, *y*	Returns a number that specifies the current location of the mouse pointer. The *x* and *y* values are always expressed in terms of the coordinate system set by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties of the object.

Remarks

Use a MouseDown or MouseUp event procedure to specify actions that will occur when a given mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events enable you to distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers.

The following applies to both Click and DblClick events:

- If a mouse button is pressed while the pointer is over a form or control, that object "captures" the mouse and receives all mouse events up to and including the last MouseUp event. This implies that the *x*, *y* mouse-pointer coordinates returned by a mouse event may not always be in the internal area of the object that receives them.
- If mouse buttons are pressed in succession, the object that captures the mouse after the first press receives all mouse events until all buttons are released.

If you need to test for the *button* or *shift* arguments, you can use constants listed in the Visual Basic (VB) object library in the Object Browser to define the bits within the argument:

Constant (Button)	Value	Description
vbLeftButton	1	Left button is pressed
vbRightButton	2	Right button is pressed
vbMiddleButton	4	Middle button is pressed

Constant (Shift)	Value	Description
vbShiftMask	1	SHIFT key is pressed.
vbCtrlMask	2	CTRL key is pressed.
vbAltMask	4	ALT key is pressed.

The constants then act as bit masks you can use to test for any combination of buttons without having to figure out the unique bit field value for each combination.

Note You can use a MouseMove event procedure to respond to an event caused by moving the mouse. The *button* argument for MouseDown and MouseUp differs from the *button* argument used for MouseMove. For MouseDown and MouseUp, the *button* argument indicates exactly one button per event, whereas for MouseMove, it indicates the current state of all buttons.

MouseIcon Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMouseIconActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMouseIconActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMouseIconActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMouseIconActiveXControlsS"}

Returns or sets a custom mouse icon.

Syntax

object.**MouseIcon = LoadPicture(***pathname***)**
object.**MouseIcon [=** *picture***]**

The **MouseIcon** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
pathname	A string expression specifying the path and filename of the file containing the custom icon.
picture	The **Picture** property of a **Form** object, **PictureBox** control, or **Image** control.

Remarks

The **MouseIcon** property provides a custom icon that is used when the **MousePointer** property is set to 99.

Although Visual Basic does not create or support color cursor (.cur) files (such as those that ship with Windows NT), you can use the **MouseIcon** property to load either cursor or icon files. Color cursor files such as those shipped with Windows NT 3.51, are displayed in black and white. To display a color cursor, use a color icon file (.ico). The **MouseIcon** property provides your program with easy access to custom cursors of any size, with any desired hot spot location. Visual Basic does not load animated cursor (.ani) files, even though 32-bit versions of Windows support these cursors.

MouseMove Event (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtMouseMoveActiveXControlsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtMouseMoveActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtMouseMoveActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtMouseMoveActiveXControlsS"}

Occurs when the user moves the mouse.

Syntax

Private Sub Form_MouseMove(*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub MDIForm_MouseMove(*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**
Private Sub *object***_MouseMove(**[*index* **As Integer**,] *button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**

The MouseMove event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
button	An integer that corresponds to the state of the mouse buttons in which a bit is set if the button is down. The *button* argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. It indicates the complete state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are pressed.
shift	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys. A bit is set if the key is down. The *shift* argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The *shift* argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of *shift* would be 6.
x, *y*	A number that specifies the current location of the mouse pointer. The *x* and *y* values are always expressed in terms of the coordinate system set by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties of the object.

Remarks

The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders.

If you need to test for the *button* or *shift* arguments, you can use constants listed in the Visual Basic (VB) object library in the Object Browser to define the bits within the argument:

Constant (Button)	Value	Description
vbLeftButton	1	Left button is pressed.
vbRightButton	2	Right button is pressed.

vbMiddleButton	4	Middle button is pressed.

Constant (Shift)	Value	Description
vbShiftMask	1	SHIFT key is pressed.
vbCtrlMask	2	CTRL key is pressed.
vbAltMask	4	ALT key is pressed.

The constants then act as bit masks you can use to test for any combination of buttons without having to figure out the unique bit field value for each combination.

You test for a condition by first assigning each result to a temporary integer variable and then comparing the *button* or *shift* arguments to a bit mask. Use the **And** operator with each argument to test if the condition is greater than zero, indicating the key or button is pressed, as in this example:

```
LeftDown = (Button And vbLeftButton) > 0
CtrlDown = (Shift And vbCtrlMask) > 0
```

Then, in a procedure, you can test for any combination of conditions, as in this example:

```
If LeftDown And CtrlDown Then
```

Note You can use MouseDown and MouseUp event procedures to respond to events caused by pressing and releasing mouse buttons.

The *button* argument for MouseMove differs from the *button* argument for MouseDown and MouseUp. For MouseMove, the *button* argument indicates the current state of all buttons; a single MouseMove event can indicate that some, all, or no buttons are pressed. For MouseDown and MouseUp, the *button* argument indicates exactly one button per event.

Any time you move a window inside a MouseMove event, it can cause a cascading event. MouseMove events are generated when the window moves underneath the pointer. A MouseMove event can be generated even if the mouse is perfectly stationary.

MousePointer Constants (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbidxMousePointerConstantsWindowsCommonControls;vbproBooksOnlineJumpTopic"}

Constant	Value	Description
ccDefault	0	(Default) Shape determined by the object.
ccArrow	1	Arrow.
ccCross	2	Cross (cross-hair pointer).
ccIbeam	3	I Beam.
ccIcon	4	Icon (small square within a square).
ccSize	5	Size (four-pointed arrow pointing north, south, east, and west).
ccSizeNESW	6	Size NE SW (double arrow pointing northeast and southwest).
ccSizeNS	7	Size N S (double arrow pointing north and south).
ccSizeNWSE	8	Size NW, SE.
ccSizeEW	9	Size E W (double arrow pointing east and west).
ccUpArrow	10	Up Arrow.
ccHourglass	11	Hourglass (wait).
ccNoDrop	12	No Drop.
ccArrowHourglass	13	Arrow and hourglass.
cc ArrowQuestion	14	Arrow and question mark.
ccSizeAll	15	Size all.
ccCustom	99	Custom icon specified by the **MouseIcon** property.

Note The cc prefix refers to the custom controls. Prefixes for the constants change with the specific control or group of controls. However, the description remains the same unless indicated.

MousePointer Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMousePointerActiveXControlsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMousePointerActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMousePointerActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMousePointerActiveXControlsS"}

Returns or sets a value indicating the type of mouse pointer displayed when the mouse is over a particular part of an object at run time.

Syntax

object.**MousePointer** [**=** *value*]

The **MousePointer** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer specifying the type of mouse pointer displayed, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
vbDefault	0	(Default) Shape determined by the object.
VbArrow	1	Arrow.
VbCrosshair	2	Cross (crosshair pointer).
VbIbeam	3	I beam.
VbIconPointer	4	Icon (small square within a square).
VbSizePointer	5	Size (four-pointed arrow pointing north, south, east, and west).
VbSizeNESW	6	Size NE SW (double arrow pointing northeast and southwest).
VbSizeNS	7	Size N S (double arrow pointing north and south).
VbSizeNWSE	8	Size NW SE (double arrow pointing northwest and southeast).
VbSizeWE	9	Size W E (double arrow pointing west and east).
VbUpArrow	10	Up Arrow.
VbHourglass	11	Hourglass (wait).
VbNoDrop	12	No Drop.
VbArrowHourglass	13	Arrow and hourglass. (Only available in 32-bit Visual Basic.)
vbArrowQuestion	14	Arrow and question mark. (Only available in 32-bit Visual Basic.)
vbSizeAll	15	Size all. (Only available in 32-bit Visual Basic.)
vbCustom	99	Custom icon specified by the **MouseIcon** property.

Remarks

You can use this property when you want to indicate changes in functionality as the mouse pointer passes over controls on a form or dialog box. The Hourglass setting (11) is useful for indicating that the user should wait for a process or operation to finish.

Note If your application calls DoEvents, the **MousePointer** property may temporarily change when over a custom control.

Key Code Constants (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstKeyCodeConstantsActiveXControlsC;vbproBooksOnlineJumpTopic"}

Key Codes

Constant	Value	Description
vbKeyLButton	&H1	Left mouse button
vbKeyRButton	&H2	Right mouse button
vbKeyCancel	&H3	CANCEL key
vbKeyMButton	&H4	Middle mouse button
vbKeyBack	&H8	BACKSPACE key
vbKeyTab	&H9	TAB key
vbKeyClear	&HC	CLEAR key
vbKeyReturn	&HD	ENTER key
vbKeyShift	&H10	SHIFT key
vbKeyControl	&H11	CTRL key
vbKeyMenu	&H12	MENU key
vbKeyPause	&H13	PAUSE key
vbKeyCapital	&H14	CAPS LOCK key
vbKeyEscape	&H1B	ESC key
vbKeySpace	&H20	SPACEBAR key
vbKeyPageUp	&H21	PAGE UP key
vbKeyPageDown	&H22	PAGE DOWN key
vbKeyEnd	&H23	END key
vbKeyHome	&H24	HOME key
vbKeyLeft	&H25	LEFT ARROW key
vbKeyUp	&H26	UP ARROW key
vbKeyRight	&H27	RIGHT ARROW key
vbKeyDown	&H28	DOWN ARROW key
vbKeySelect	&H29	SELECT key
vbKeyPrint	&H2A	PRINT SCREEN key
vbKeyExecute	&H2B	EXECUTE key
vbKeySnapshot	&H2C	SNAPSHOT key
vbKeyInsert	&H2D	INS key
vbKeyDelete	&H2E	DEL key
vbKeyHelp	&H2F	HELP key
vbKeyNumlock	&H90	NUM LOCK key

KeyA Through KeyZ Are the Same as Their ASCII Equivalents: 'A' Through 'Z'

Constant	Value	Description
vbKeyA	65	A key
vbKeyB	66	B key
vbKeyC	67	C key
vbKeyD	68	D key
vbKeyE	69	E key

vbKeyF	70	F key
vbKeyG	71	G key
vbKeyH	72	H key
vbKeyI	73	I key
vbKeyJ	74	J key
vbKeyK	75	K key
vbKeyL	76	L key
vbKeyM	77	M key
vbKeyN	78	N key
vbKeyO	79	O key
vbKeyP	80	P key
vbKeyQ	81	Q key
vbKeyR	82	R key
vbKeyS	83	S key
vbKeyT	84	T key
vbKeyU	85	U key
vbKeyV	86	V key
vbKeyW	87	W key
vbKeyX	88	X key
vbKeyY	89	Y key
vbKeyZ	90	Z key

Key0 Through Key9 Are the Same as Their ASCII Equivalents: '0' Through '9

Constant	Value	Description
vbKey0	48	0 key
vbKey1	49	1 key
vbKey2	50	2 key
vbKey3	51	3 key
vbKey4	52	4 key
vbKey5	53	5 key
vbKey6	54	6 key
vbKey7	55	7 key
vbKey8	56	8 key
vbKey9	57	9 key

Keys on the Numeric Keypad

Constant	Value	Description
vbKeyNumpad0	&H60	0 key
vbKeyNumpad1	&H61	1 key
vbKeyNumpad2	&H62	2 key
vbKeyNumpad3	&H63	3 key
vbKeyNumpad4	&H64	4 key
vbKeyNumpad5	&H65	5 key
vbKeyNumpad6	&H66	6 key
vbKeyNumpad7	&H67	7 key

vbKeyNumpad8	&H68	8 key
vbKeyNumpad9	&H69	9 key
vbKeyMultiply	&H6A	MULTIPLICATION SIGN (*) key
vbKeyAdd	&H6B	PLUS SIGN (+) key
vbKeySeparator	&H6C	ENTER (keypad) key
vbKeySubtract	&H6D	MINUS SIGN (-) key
vbKeyDecimal	&H6E	DECIMAL POINT(.) key
vbKeyDivide	&H6F	DIVISION SIGN (/) key

Function Keys

Constant	Value	Description
vbKeyF1	&H70	F1 key
vbKeyF2	&H71	F2 key
vbKeyF3	&H72	F3 key
vbKeyF4	&H73	F4 key
vbKeyF5	&H74	F5 key
vbKeyF6	&H75	F6 key
vbKeyF7	&H76	F7 key
vbKeyF8	&H77	F8 key
vbKeyF9	&H78	F9 key
vbKeyF10	&H79	F10 key
vbKeyF11	&H7A	F11 key
vbKeyF12	&H7B	F12 key
vbKeyF13	&H7C	F13 key
vbKeyF14	&H7D	F14 key
vbKeyF15	&H7E	F15 key
vbKeyF16	&H7F	F16 key

Max, Min Properties (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMaxMinPropertiesActiveXControlsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMaxMinPropertiesActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMaxMinPropertiesActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMaxMinPropertiesActiveXControlsS"}

- **Max** — returns or sets a scroll bar position's maximum **Value** property setting when the scroll box is in its bottom or rightmost position. For the ProgressBar control, it returns or sets its maximum value.
- **Min** — returns or sets a scroll bar position's minimum **Value** property setting when the scroll box is in its top or leftmost position. For the ProgressBar control, it returns or sets its minimum value.

Syntax

object.**Max** [**=** *value*]
object.**Min** [**=** *value*]

The **Max** and **Min** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A numeric expression specifying the maximum or minimum **Value** property setting, as described in Settings.

Settings

For each property, you can specify an integer between -32,768 and 32,767, inclusive. The default settings are:

- **Max** — 32,767.
- **Min** — 0.

Remarks

The Microsoft Windows operating environment automatically sets ranges for scroll bars proportional to the contents of forms, **ComboBox** controls, and **ListBox** controls. For a scroll bar (**HScrollBar** or **VScrollBar)** control, however, you must specify these ranges. Use **Max** and **Min** to set a range appropriate to how the scroll bar control is used — for example, as an input device or as an indicator of speed or quantity.

Typically, you set **Max** and **Min** at design time. You can also set them in code at run time if the scrolling range must change dynamically — for example, when adding records to a database that can be scrolled through. You set the maximum and minimum scrolling increments for a scroll bar control with the **LargeChange** and **SmallChange** properties.

Note If **Max** is set to less than **Min**, the maximum value is set at the leftmost or topmost position of a horizontal or vertical scroll bar, respectively. The **Max** property of a **ProgressBar** control must always be greater than its **Min** property, and its **Min** property must always be greater than or equal to 0.

The **Max** and **Min** properties define the range of the control. The **ProgressBar** control's **Min** property is 0 and its **Max** property is 100 by default, representing the percentage duration of the operation.

OLECompleteDrag Event (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLECompleteDragEventActiveXControlsC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLECompleteDragEventActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLECompleteDragEventActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLECompleteDragEventActiveXControlsS"}

Occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled.

Syntax

Private Sub *object*_**CompleteDrag(**[*effect* **As Long**]**)**

The CompleteDrag event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
effect	A long integer set by the source object identifying the action that has been performed, thus allowing the source to take appropriate action if the component was moved (such as the source deleting data if it is moved from one component to another). The possible values are listed in Settings.

Settings

The settings for *effect* are:

Constant	Value	Description
vbDropEffectNone	0	Drop target cannot accept the data, or the drop operation was cancelled.
vbDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
vbDropEffectMove	2	Drop results in a link to the original data being created between drag source and drop target.

Remarks

The OLECompleteDrag event is the final event to be called in an OLE drag/drop operation. This event informs the source component of the action that was performed when the object was dropped onto the target component. The target sets this value through the *effect* parameter of the OLEDragDrop event. Based on this, the source can then determine the appropriate action it needs to take. For example, if the object was moved into the target (**vbDropEffectMove**), the source needs to delete the object from itself after the move.

If **OLEDragMode** is set to **Automatic**, then Visual Basic handles the default behavior. The event still occurs, however, allowing the user to add to or change the behavior.

Most components support manual OLE drag and drop events, and some support automatic OLE drag and drop events.

OLEDrag Method (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthOLEDragMethodActiveXControlsC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthOLEDragMethodActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthOLEDragMethodActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthOLEDragMethodActiveXControlsS"}

Causes a component to initiate an OLE drag/drop operation.

Syntax

object.**OLEDrag**

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

Remarks

When the **OLEDrag** method is called, the component's OLEStartDrag event occurs, allowing it to supply data to a target component.

Text Property (ActiveX Controls) Example

This example populates a **TreeView** control with the titles of files in a **ListBox** control. When an item in the **TreeView** control is clicked, the **Text** property is displayed in a **Label** on the form. To try the example, place **TreeView**, **Label**, and **ListBox** controls on a form and paste the code into the form's Declarations section. Run the example and click on any item to see its **Text** property.

```
Private Sub Form_Load()
   Dim nodX As Node  ' Declare an object variable for the Node.
   Dim i As Integer  ' Declare a variable for use as a counter.

   ' Add one Node to the TreeView control, and call it the first node
   Set nodX = TreeView1.Nodes.Add()
   nodX.Text = "First Node"

   'Populate the ListBox
      List1.AddItem "Node1"   ' Add each item to list.
      List1.AddItem "Node2"
      List1.AddItem "Node3"
      List1.AddItem "Node4"
      List1.AddItem "Node5"
      List1.AddItem "Node6"
      List1.AddItem "Node7"

   ' Add child nodes to the first Node object. Use the
   ' ListBox to populate the control.
   For i = 0 To List1.ListCount - 1
      Set nodX = TreeView1.Nodes.Add(1, tvwChild)
      nodX.Text = List1.List(i)
   Next i
   Treeview1.Nodes(1).Selected = True
   nodX.EnsureVisible   ' Make sure the node is visible.
End Sub

Private Sub TreeView1_NodeClick(ByVal Node As Node)
   ' Display the clicked Node object's Text property.
   Label1.Caption = Node.Text
End Sub
```

OLEDragDrop Event (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEDragDropEventActiveXControlsC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEDragDropEventActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEDragDropEventActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEDragDropEventActiveXControlsS"}

Occurs when a source component is dropped onto a target component when the source component determines that a drop can occur.

Note This event occurs only if **OLEDropMode** is set to **1** (**Manual**).

Syntax

Private Sub *object*_**OLEDragDrop(***data* **As DataObject**, *effect* **As Long**, *button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**

The OLEDragDrop event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
data	A **DataObject** object containing formats that the source will provide and, in addition, possibly the data for those formats. If no data is contained in the **DataObject**, it is provided when the control calls the **GetData** method. The **SetData** and **Clear** methods cannot be used here.
effect	A long integer set by the target component identifying the action that has been performed (if any), thus allowing the source to take appropriate action if the component was moved (such as the source deleting the data). The possible values are listed in Settings.
button	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively. It indicates the state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are depressed.
shift	An integer which acts as a bit field corresponding to the state of the SHIFT, CTRL, and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The *shift* parameter indicates the state of these keys; some, all, or none of the bits can be set, indicating that some, all, or none of the keys are depressed. For example, if both the CTRL and ALT keys were depressed, the value of *shift* would be 6.
x,y	A number which specifies the current location of the mouse pointer. The *x* and *y* values are always expressed in terms of the coordinate system set by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties of the object.

Settings

The settings for *effect* are:

Constant	Value	Description
vbDropEffectNone	0	Drop target cannot accept the data.

vbDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
vbDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

Remarks

The source ActiveX component should always mask values from the *effect* parameter to ensure compatibility with future implementations of ActiveX components. Presently, only three of the 32 bits in the *effect* parameter are used. In future versions of Visual Basic, however, these other bits may be used. Therefore, as a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.

For example, a source component should not compare an *effect* against, say, **vbDropEffectCopy**, such as in this manner:

```
If Effect = vbDropEffectCopy...
```

Instead, the source component should mask for the value or values being sought, such as this:

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```

-or-

```
If (Effect And vbDropEffectCopy)...
```

This allows for the definition of new drop effects in future versions of Visual Basic while preserving backwards compatibility with your existing code.

Most components support manual OLE drag and drop events, and some support automatic OLE drag and drop events.

OLEDragMode Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOLEDragModePropertyActiveXControlsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDragModePropertyActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDragModePropertyActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDragModePropertyActiveXControlsS"}

Returns or sets whether the component or the programmer handles an OLE drag/drop operation.

Syntax

object.**OLEDragMode** = *mode*

The **OLEDragMode** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
mode	An integer which specifies the method with which an component handles OLE drag/drop operations, as described in Settings.

Settings

The settings for *mode* are:

Constant	Value	Description
vbOLEDragManual	0	(Default) Manual. The programmer handles all OLE drag/drop operations.
vbOLEDragAutomatic	1	Automatic. The component handles all OLE drag/drop operations.

Remarks

When **OLEDragMode** is set to **Manual**, you must call the **OLEDrag** method to start dragging, which then triggers the **OLEStartDrag** event.

When **OLEDragMode** is set to **Automatic**, the source component fills the **DataObject** object with the data it contains and sets the *effects* parameter before initiating the OLEStartDrag event (as well as the OLESetData and other source-level OLE drag/drop events) when the user attempts to drag out of the control. This gives you control over the drag/drop operation and allows you to intercede by adding other formats, or by overriding or disabling the automatic data and formats using the **Clear** or **SetData** methods.

If the source's **OLEDragMode** property is set to **Automatic**, and no data is loaded in the **OLEStartDrag** event, or *aftereffects* is set to **0**, then the OLE drag/drop operation does not occur.

Note If the **DragMode** property of a control is set to **Automatic**, the setting of **OLEDragMode** is ignored, because regular Visual Basic drag and drop events take precedence.

OLEDragOver Event (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEDragOverEventActiveXControlsC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEDragOverEventActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEDragOverEventActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEDragOverEventActiveXControlsS"}

Occurs when one component is dragged over another.

Syntax

Private Sub *object*_**OLEDragOver(***data* **As DataObject**, *effect* **As Long**, *button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single**, *state* **As Integer)**

The OLEDragOver event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
data	A **DataObject** object containing formats that the source will provide and, in addition, possibly the data for those formats. If no data is contained in the **DataObject**, it is provided when the control calls the **GetData** method. The **SetData** and **Clear** methods cannot be used here.
effect	A long integer initially set by the source object identifying all effects it supports. This parameter must be correctly set by the target component during this event. The value of *effect* is determined by logically **Or**'ing together all active effects (as listed in Settings). The target component should check these effects and other parameters to determine which actions are appropriate for it, and then set this parameter to one of the allowable effects (as specified by the source) to specify which actions will be performed if the user drops the selection on the component. The possible values are listed in Settings.
button	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively. It indicates the state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are depressed.
shift	An integer which acts as a bit field corresponding to the state of the SHIFT, CTRL, and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The *shift* parameter indicates the state of these keys; some, all, or none of the bits can be set, indicating that some, all, or none of the keys are depressed. For example, if both the CTRL and ALT keys are depressed, the value of *shift* would be 6.
x,y	A number that specifies the current horizontal (*x*) and vertical (*y*) position of the mouse pointer within the target form or control. The *x* and *y* values are always expressed in terms of the coordinate system set by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties of the object.
state	An integer that corresponds to the transition state of the

control being dragged in relation to a target form or control. The possible values are listed in Settings.

Settings

The settings for *effect* are:

Constant	Value	Description
vbDropEffectNone	0	Drop target cannot accept the data.
vbDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
vbDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.
vbDropEffectScroll	-2147483648 (&H80000000)	Scrolling is occuring or about to occur in the target component. This value is used in conjunction with the other values. **Note** Use only if you are performing your own scrolling in the target component.

The settings for *state* are:

Constant	Value	Description
vbEnter	0	Source component is being dragged within the range of a target.
vbLeave	1	Source component is being dragged out of the range of a target.
vbOver	2	Source component has moved from one position in the target to another.

Remarks

Note If the *state* parameter is **vbLeave**, indicating that the mouse pointer has left the target, then the *x* and *y* parameters will contain zeros.

The source component should always mask values from the *effect* parameter to ensure compatibility with future implementations of ActiveX components. Presently, only three of the 32 bits in the *effect* parameter are used. In future versions of Visual Basic, however, these other bits may be used. Therefore, as a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.

For example, a source component should not compare an *effect* against, say, **vbDropEffectCopy**, such as in this manner:

```
If Effect = vbDropEffectCopy...
```

Instead, the source component should mask for the value or values being sought, such as this:

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```

-or-

```
If (Effect And vbDropEffectCopy)...
```

This allows for the definition of new drop effects in future versions of Visual Basic while preserving backwards compatibility with your existing code.

Most components support manual OLE drag and drop events, and some support automatic OLE drag and drop events.

OLEDropMode Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOLEDropModePropertyActiveXControlsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDropModePropertyActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDropModePropertyActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDropModePropertyActiveXControlsS"}

Returns or sets how a target component handles drop operations.

Syntax

object.**OLEDropMode** [= *mode*]

The **OLEDropMode** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
mode	An enumerated integer which specifies the method which a component handles OLE drag/drop operations, as described in Settings.

Settings

The settings for *mode* are:

Constant	Value	Description
vbOLEDropNone	0	(Default) None. The target component does not accept OLE drops and displays the No Drop cursor.
vbOLEDropManual	1	Manual. The target component triggers the OLE drop events, allowing the programmer to handle the OLE drop operation in code.
vbOLEDropAutomatic	2	Automatic. The target component automatically accepts OLE drops if the **DataObject** object contains data in a format it recognizes. No mouse or OLE drag/drop events on the target will occur when **OLEDropMode** is set to **vbOLEDropAutomatic**.

Remarks

Note The target component inspects what is being dragged over it in order to determine which events to trigger; the OLE drag/drop events, or the Visual Basic drag/drop events. There is no collision of components or confusion about which events are fired, since only one type of object can be dragged at a time.

OLEGiveFeedback Event (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEGiveFeedbackEventActiveXControlsC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEGiveFeedbackEventActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEGiveFeedbackEventActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEGiveFeedbackEventActiveXControlsS"}

Occurs after every OLEDragOver event. OLEGiveFeedback allows the source component to provide visual feedback to the user, such as changing the mouse cursor to indicate what will happen if the user drops the object, or provide visual feedback on the selection (in the source component) to indicate what will happen.

Syntax

Private Sub *object*_**OLEGiveFeedback(***effect* **As Long**, defaultcursors **As Boolean)**

The OLEGiveFeedback event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
effect	A long integer set by the target component in the OLEDragOver event specifying the action to be performed if the user drops the selection on it. This allows the source to take the appropriate action (such as giving visual feedback). The possible values are listed in Settings.
defaultcursors	A boolean value which determines whether Visual Basic uses the default mouse cursor proved by the component, or uses a user-defined mouse cursor. **True** (default) = use default mouse cursor. **False** = do not use default cursor. Mouse cursor must be set with the **MousePointer** property of the **Screen** object.

Settings

The settings for *effect* are:

Constant	Value	Description
vbDropEffectNone	0	Drop target cannot accept the data.
vbDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
vbDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.
vbDropEffectScroll	-2147483648 (&H80000000)	Scrolling is occuring or about to occur in the target component. This value is used in conjunction with the other values. **Note** Use only if you are performing your own scrolling in the target

component.

Remarks

If there is no code in the OLEGiveFeedback event, or if the *defaultcursors* parameter is set to **True**, then Visual Basic automatically sets the mouse cursor to the default cursor provided by the component.

The source component should always mask values from the *effect* parameter to ensure compatibility with future implementations of components. Presently, only three of the 32 bits in the *effect* parameter are used. In future versions of Visual Basic, however, these other bits may be used. Therefore, as a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.

For example, a source component should not compare an *effect* against, say, **vbDropEffectCopy**, such as in this manner:

```
If Effect = vbDropEffectCopy...
```

Instead, the source component should mask for the value or values being sought, such as this:

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```

-or-

```
If (Effect And vbDropEffectCopy)...
```

This allows for the definition of new drop effects in future versions of Visual Basic while preserving backwards compatibility with your existing code.

Most components support manual OLE drag and drop events, and some support automatic OLE drag and drop events.

OLESetData Event (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLESetDataEventActiveXControlsC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLESetDataEventActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLESetDataEventActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLESetDataEventActiveXControlsS"}

Occurs on an source component when a target component performs the **GetData** method on the source's **DataObject** object, but the data for the specified format has not yet been loaded.

Syntax

Private Sub *object*_**OLESetData(***data* **As DataObject**, *dataformat* **As Integer)**

The OLESetData event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
data	A **DataObject** object in which to place the requested data. The component calls the **SetData** method to load the requested format.
dataformat	An integer specifying the format of the data that the target component is requesting. The source component uses this value to determine what to load into the **DataObject** object.

Remarks

In certain cases, you may wish to defer loading data into the **DataObject** object of a source component to save time, especially if the source component supports many formats. This event allows the source to respond to only one request for a given format of data. When this event is called, the source should check the *format* parameter to determine what needs to be loaded and then perform the **SetData** method on the **DataObject** object to load the data which is then passed back to the target component.

OLEStartDrag Event (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEStartDragEventActiveXControlsC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEStartDragEventActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEStartDragEventActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEStartDragEventActiveXControlsS"}

Occurs when a component's **OLEDrag** method is performed, or when a component initiates an OLE drag/drop operation when the **OLEDragMode** property is set to **Automatic**.

This event specifies the data formats and drop effects that the source component supports. It can also be used to insert data into the **DataObject** object.

Syntax

Private Sub *object*_**StartDrag(***data* **As DataObject**, *allowedeffects* **As Long)**

The StartDrag event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
data	A **DataObject** object containing formats that the source will provide and, optionally, the data for those formats. If no data is contained in the **DataObject**, it is provided when the control calls the **GetData** method. The programmer should provide the values for this parameter in this event. The **SetData** and **Clear** methods cannot be used here.
allowedeffects	A long integer containing the effects that the source component supports. The possible values are listed in Settings. The programmer should provide the values for this parameter in this event.

Settings

The settings for *allowedeffects* are:

Constant	Value	Description
vbDropEffectNone	0	Drop target cannot accept the data.
vbDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
vbDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

Remarks

The source component should logically **Or** together the supported values and places the result in the *allowedeffects* parameter. The target component can use this value to determine the appropriate action (and what the appropriate user feedback should be).

The StartDrag event also occurs if the component's **OLEDragMode** property is set to **Automatic**. This allows you to add formats and data to the **DataObject** object after the component has done so. You can also override the default behavior of the component by clearing the **DataObject** object (using the **Clear** method) and then adding your data and formats.

You may wish to defer putting data into the **DataObject** object until the target component requests it.

This allows the source component to save time by not loading multiple data formats. When the target performs the **GetData** method on the **DataObject**, the source's OLESetData event will occur if the requested data is not contained in the **DataObject**. At this point, the data can be loaded into the **DataObject**, which will in turn provide the data to the target.

If the user does not load any formats into the **DataObject**, then the drag/drop operation is canceled.

SetData Method (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSetDataMethodActiveXControlsC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSetDataMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSetDataMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSetDataMethodS"}

Inserts data into a **DataObject** object using the specified data format.

Syntax

object.**SetData [***data***]**, **[***format***]**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
data	Optional A variant containing the data to be passed to the **DataObject** object.
format	Optional. A constant or value that specifies the format of the data being passed, as described in Settings.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFText	1	Text (.txt files)
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	Metafile (.wmf files)
vbCFEMetafile	14	Enhanced metafile (.emf files)
vbCFDIB	8	Device-independent bitmap (DIB)
vbCFPalette	9	Color palette
vbCFFiles	15	List of files
vbCFRTF	-16639	Rich text format (.rtf files)

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

The *data* argument is optional. This allows you to set several different formats that the source component can support without having to load the data separately for each format. Multiple formats are set by calling **SetData** several times, each time using a different format. If you wish to start fresh, use the **Clear** method to clear all data and format information from the **DataObject**.

The *format* argument is also optional, but either the *data* or *format* argument must be specified. If *data* is specified, but not *format*, then Visual Basic will try to determine the format of the data. If it is unsuccessful, then an error is generated. When the target requests the data, and a format was specified, but no data was provided, the source's **OLESetData** event occurs, and the source can then provide the requested data type.

It's possible for the **GetData** and **SetData** methods to use data formats other than those listed in Settings, including user-defined formats registered with Windows via the `RegisterClipboardFormat()` API function. However, there are a few caveats:

- The **SetData** method requires the data to be in the form of a byte array when it does not recognize the data format specified.
- The **GetData** method always returns data in a byte array when it is in a format that it doesn't

recognize, although Visual Basic can transparently convert this returned byte array into other data types, such as strings.

- The byte array returned by **GetData** will be larger than the actual data when running on some operating systems, with arbitrary bytes at the end of the array. The reason for this is that Visual Basic does not know the data's format, and knows only the amount of memory that the operating system has allocated for the data. This allocation of memory is often larger than is actually required for the data. Therefore, there may be extraneous bytes near the end of the allocated memory segment. As a result, you must use appropriate functions to interpret the returned data in a meaningful way (such as truncating a string at a particular length with the **Left** function if the data is in a text format).

Picture Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPictureActiveXControlsC;vbproPictureC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPictureActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPictureActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPictureActiveXControlsS"}

Returns or sets a graphic to be displayed in a control. For the **OLE** container control, not available at design time and read-only at run time.

Syntax

object.**Picture** [= *picture*]

The **Picture** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
picture	A string expression specifying a file containing a graphic, as described in Settings.

Settings

The settings for *picture* are:

Setting	Description
(None)	(Default) No picture.
(Bitmap, icon, metafile)	Specifies a graphic. You can load the graphic from the Properties window at design time. At run time, you can also set this property using the **LoadPicture** function on a bitmap, icon, or metafile.

Remarks

At design time, you can transfer a graphic with the Clipboard using the Copy, Cut, and Paste commands on the Edit menu. At run time, you can use Clipboard methods such as **GetData**, **SetData**, and **GetFormat** with the nontext Clipboard constants **vbCFBitmap**, **vbCFMetafile**, and **vbCFDIB**, which are listed in the Visual Basic (VB) object library in the Object Browser.

When setting the **Picture** property at design time, the graphic is saved and loaded with the form. If you create an executable file, the file contains the image. When you load a graphic at run time, the graphic isn't saved with the application. Use the **SavePicture** statement to save a graphic from a form or picture box into a file.

Note At run time, the **Picture** property can be set to any other object's **DragIcon**, **Icon**, **Image**, or **Picture** property, or you can assign it the graphic returned by the **LoadPicture** function. The exception to this is the **Picture** property of the **ListImages** object, which is a read-only property.

SelLength, SelStart, SelText Properties (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSelLengthC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelLengthX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelLengthA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelLengthS"}

- **SelLength** — returns or sets the number of characters selected.
- **SelStart** — returns or sets the starting point of text selected; indicates the position of the insertion point if no text is selected.
- **SelText** — returns or sets the string containing the currently selected text; consists of a zero-length string ("") if no characters are selected.

These properties aren't available at design time.

Syntax

object.**SelLength** [**=** *number*]
object.**SelStart** [**=** *index*]
object.**SelText** [**=** *value*]

The **SelLength**, **SelStart**, and **SelText** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression specifying the number of characters selected. For **SelLength** and **SelStart**, the valid range of settings is 0 to text length — the total number of characters in the edit area of a **ComboBox** or **TextBox** control.
index	A numeric expression specifying the starting point of the selected text, as described in Settings.
value	A string expression containing the selected text.

Remarks

Use these properties for tasks such as setting the insertion point, establishing an insertion range, selecting substrings in a control, or clearing text. Used in conjunction with the **Clipboard** object, these properties are useful for copy, cut, and paste operations.

When working with these properties:

- Setting **SelLength** less than 0 causes a run-time error.
- Setting **SelStart** greater than the text length sets the property to the existing text length; changing **SelStart** changes the selection to an insertion point and sets **SelLength** to 0.
- Setting **SelText** to a new value sets **SelLength** to 0 and replaces the selected text with the new string.

ShowTips Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTooltipsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTooltipsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTooltipsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTooltipsS"}

Returns a value that determines whether ToolTips are displayed for an object.

Syntax

object.**ShowTips** [= *value*]

The **ShowTips** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression specifying whether ToolTips are displayed, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	(Default) Each object in the control may display an associated string, which is the setting of the **ToolTipText** property, in a small rectangle below the object. This ToolTip appears when the user's cursor hovers over the object at run time for about one second.
False	An object will not display a ToolTip at run time .

Remarks

At design time you can set the **ShowTips** property on the General tab in the control's Property Pages dialog box.

Text Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTextActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTextActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTextActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTextActiveXControlsS"}

Returns or sets the text contained in an object.

Syntax

object.**Text** [= *string*]

The **Text** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression specifying the text appearing in the object.

Value Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproValueActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproValueActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproValueActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproValueActiveXControlsS"}

Returns or sets the value of an object.

Syntax

object.**Value** [= *integer*]

The **Value** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
integer	For a **Slider** control, a long integer that specifies the current position of the slider. For the **ProgressBar** control, an integer that specifies the value of the **ProgressBar** control. For other controls, see Settings below.

Settings

For the **Button** object, the settings for *integer* are:

Constant	Value	Description
tbrUnPressed	0	(Default). The button is not currently pressed or checked.
tbrPressed	1	The button is currently pressed or checked.

Remarks

- **Slider** control—returns or sets the current position of the slider. **Value** is always between the values for the **Max** and **Min** properties, inclusive, for a **Slider** control.
- **ProgressBar** control—returns or sets a value indicating an operation's approximate progress toward completion. Incrementing the **Value** property doesn't change the appearance of the **ProgressBar** control by the exact value of the **Value** property. **Value** is always in the range

between the values for the **Max** and **Min** properties, inclusive. Not available at design time.

Value Property (ActiveX Controls) Example

This example uses the **Value** property to determine which icon from an associated **ImageList** control is displayed on the **Toolbar** control. To try the example, place a **Toolbar** control on a form and paste the code into the form's Declarations section. Then run the example.

```
Private Sub Toolbar1_ButtonClick(ByVal Button As Button)
   ' Use the Key value to determine which button has been clicked.
   Select Case Button.Key

   Case "Done" ' A check button.
      If Button.Value = vbUnchecked Then
         ' The button is unchecked.
         Button.Value = vbChecked  ' Check the button.
         ' Assuming there is a ListImage object with
         ' key "down."
         Button.Image = "down"
      Else  ' Uncheck the button
         Button.Value = vbUnchecked
         ' Assuming there is a ListImage object with
         ' key " up."
         Button.Image = "up"
      End If

   ' More Cases are possible.
   End Select
End Sub
```

RemoteHost Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproInetRemoteHostC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproInetRemoteHostX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproInetRemoteHostA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproInetRemoteHostS"}

Returns or sets the remote machine to which a control sends or receives data. You can either provide a host name, for example, "FTP://ftp.microsoft.com," or an IP address string in dotted format, such as "100.0.1.1".

Syntax

object.**RemoteHost** = *string*

The **RemoteHost** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	The name or address of the remote computer.

Data Type

String

Remarks

When this property is specified, the **URL** property is updated to show the new value. Also, if the host portion of the URL is updated, this property is also updated to reflect the new value.

The **RemoteHost** property can also be changed when invoking the **OpenURL** or **Execute** methods.

At run time, changing this value has no effect until the next connection.

RemotePort Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproInetRemotePortC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproInetRemotePortX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproInetRemotePortA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproInetRemotePortS"}

Returns or sets the remote port number to connect to.

Syntax

*object.***RemotePort** = *port*

The **RemotePort** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
port	The port to connect to. The default value of this property is 80.

Data Type

Long

Remarks

When you set the **Protocol** property, the **RemotePort** property is set automatically to the appropriate default port for each protocol. Default port numbers are shown in the table below:

Port	Description
80	HTTP, commonly used for WorldWideWeb connections.
21	FTP.

hDC Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHDCPropertyActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHDCPropertyActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHDCPropertyActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHDCPropertyActiveXControlsS"}

Returns a handle provided by the Microsoft Windows operating environment to the device context of an object.

Syntax

object.**hDC**

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

Remarks

This property is a Windows operating environment device context handle. The Windows operating environment manages the system display by assigning a device context for the **Printer** object and for each form and **PictureBox** control in your application. You can use the **hDC** property to refer to the handle for an object's device context. This provides a value to pass to Windows API calls.

With a **CommonDialog** control, this property returns a device context for the printer selected in the Print dialog box when the **cdlReturnDC** flag is set or an information context when the **cdlReturnIC** flag is set.

Note The value of the **hDC** property can change while a program is running, so don't store the value in a variable; instead, use the **hDC** property each time you need it.

The **AutoRedraw** property can cause the **hDC** property setting to change. If **AutoRedraw** is set to **True** for a form or **PictureBox** container, **hDC** acts as a handle to the device context of the persistent graphic (equivalent to the **Image** property). When **AutoRedraw** is **False**, **hDC** is the actual **hDC** value of the Form window or the **PictureBox** container. The **hDC** property setting may change while the program is running regardless of the **AutoRedraw** setting.

Height, Width Properties (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHeightWidthPropertiesActiveXControlsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHeightWidthPropertiesActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHeightWidthPropertiesActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHeightWidthPropertiesActiveXControlsS"}

Return or set the dimensions of an object or the width of the **Columns** object of a **DBGrid** control. For the **Printer** and **Screen** objects, not available at design time.

Syntax

object.**Height** [= *number*]
object.**Width** [= *number*]

The **Height** and **Width** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression specifying the dimensions of an object, as described in Settings.

Settings

Measurements are calculated as follows:

- **Form** — the external height and width of the form, including the borders and title bar.
- **Control** — measured from the center of the control's border so that controls with different border widths align correctly. These properties use the scale units of a control's container.
- **Printer** object — the physical dimensions of the paper set up for the printing device; not available at design time. If set at run time, values in these properties are used instead of the setting of the **PaperSize** property.
- **Screen** object — the height and width of the screen; not available at design time and read-only at run time.
- **Picture** object — the height and width of the picture in HiMetric units.

Remarks

For **Form**, **Printer**, and **Screen** objects, these properties are always measured in twips. For a form or control, the values for these properties change as the object is sized by the user or by your code. Maximum limits of these properties for all objects are system-dependent.

If you set the **Height** and **Width** properties for a printer driver that doesn't allow these properties to be set, no error occurs and the size of the paper remains as it was. If you set **Height** and **Width** for a printer driver that allows only certain values to be specified, no error occurs and the property is set to whatever the driver allows. For example, you could set **Height** to 150 and the driver would set it to 144.

Use the **Height**, **Width**, **Left**, and **Top** properties for operations or calculations based on an object's total area, such as sizing or moving the object. Use the **ScaleLeft**, **ScaleTop**, **ScaleHeight**, and **ScaleWidth** properties for operations or calculations based on an object's internal area, such as drawing or moving objects within another object.

Note The **Height** property can't be changed for the **DriveListBox** control or for the **ComboBox** control, whose **Style** property setting is 0 (Dropdown Combo) or 2 (Dropdown List).

For the **Columns** object of the **DBGrid** control, **Width** is specified in the unit of measure of the object that contains the **DBGrid**. The default value for **Width** is the value of the **DefColWidth** property of **DBGrid**.

For the **Picture** object, use the **ScaleX** and **ScaleY** methods to convert HiMetric units into the scale you need.

Index Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproIndexPropertyActiveXControlsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIndexPropertyActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIndexPropertyActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIndexPropertyActiveXControlsS"}

Returns or sets the number that uniquely identifies an object in a collection.

Syntax

object.**Index**

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

Remarks

The **Index** property is set by default to the order of the creation of objects in a collection. The index for the first object in a collection will always be one.

The value of the **Index** property of an object can change when objects in the collection are reordered, such as when you set the **Sorted** property to **True**. If you expect the **Index** property to change dynamically, it may be more useful to refer to objects in a collection by using the **Key** property.

Left, Top Properties (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLeftTopPropertiesActiveXControlsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLeftTopPropertiesActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLeftTopPropertiesActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLeftTopPropertiesActiveXControlsS"}

- **Left** — returns or sets the distance between the internal left edge of an object and the left edge of its container.
- **Top** — returns or sets the distance between the internal top edge of an object and the top edge of its container.
- For the **Panel** object only, this is a read-only property.

Syntax

object.**Left** [= *value*]
object.**Top** [= *value*]

The **Left** and **Top** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A numeric expression specifying distance.

Remarks

For a form, the **Left** and **Top** properties are always expressed in twips; for a control, they are measured in units depending on the coordinate system of its container. The values for these properties change as the object is moved by the user or by code. For the **CommonDialog** and **Timer** controls, these properties aren't available at run time.

For either property, you can specify a single-precision number.

Use the **Left**, **Top**, **Height**, and **Width** properties for operations based on an object's external dimensions, such as moving or resizing. Use the **ScaleLeft**, **ScaleTop**, **ScaleHeight**, and **ScaleWidth** properties for operations based on an object's internal dimensions, such as drawing or moving objects that are contained within the object. The scale-related properties apply only to **PictureBox** controls and **Form** and **Printer** objects.

Remove Method (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRemoveMethodActiveXControlsC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRemoveMethodActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRemoveMethodActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRemoveMethodActiveXControlsS"}

Removes a specific member from a collection.

Syntax

object.**Remove** *index*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer or string that uniquely identifies the object within the collection. Use an integer to specify the value of the **Index** property; use a string to specify the value of the **Key** property.

Remarks

To remove all the members of a collection, use the **Clear** method.

Tag Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTagPropertyActiveXControlsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTagPropertyActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTagPropertyActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTagPropertyActiveXControlsS"}

Returns or sets an expression that stores any extra data needed for your program. Unlike other properties, the value of the **Tag** property isn't used by Visual Basic; you can use this property to identify objects.

Syntax

object.**Tag** [= *expression*]

The **Tag** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
expression	A string expression identifying the object. The default is a zero-length string ("").

Remarks

You can use this property to assign an identification string to an object without affecting any of its other property settings or causing side effects. The **Tag** property is useful when you need to check the identity of a control or **MDIForm** object that is passed as a variable to a procedure.

Tip When you create a new instance of a form, assign a unique value to the **Tag** property.

Note The **Tag** property is of type Variant for ActiveX control collections such as **Toolbar Button** objects, **TreeView Node** objects, **ListView ListItem** and **ColumnHeader** objects, **ImageList ListImage** objects, **TabStrip Tab** objects, and **StatusBar Panel** objects. This is a powerful language feature that enables you to pass, for example, objects (such as a selected **Node**) as a **Tag**.

Visible Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproVisiblePropertyActiveXControlsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproVisiblePropertyActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproVisiblePropertyActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproVisiblePropertyActiveXControlsS"}

Returns or sets a value indicating whether an object is visible or hidden.

Syntax

object.**Visible** [= *boolean*]

The **Visible** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying whether the object is visible or hidden.

Settings

The settings for *boolean* are:

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

Remarks

To hide an object at startup, set the **Visible** property to **False** at design time. Setting this property in code enables you to hide and later redisplay a control at run time in response to a particular event.

Note Using the **Show** or **Hide** method on a form is the same as setting the form's **Visible** property in code to **True** or **False**, respectively.

Object Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproObjectPropertyActiveXControlsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproObjectPropertyActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproObjectPropertyActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproObjectPropertyActiveXControlsS"}

Returns the object and/or a setting of an object's method or property.

Syntax

object.**Object**[.*property* | .*method*]

The **Object** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
property	Property that the object supports.
method	Method that the object supports.

Remarks

Use this property to specify an object you want to use in an Automation task.

You use the object returned by the **Object** property in an Automation task by using the properties and methods of that object. For information on which properties and methods an object supports, see the documentation for the application that created the object.

Property Pages (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPropertyPagesActiveXControlsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPropertyPagesActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPropertyPagesActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPropertyPagesActiveXControlsS"}

To get Help for the properties contained in a control's **Property Page** dialog box, click the button below corresponding to the control in which you are interested.Then click the property name to get Help for that property.

{button ,JI(`',`propAnimation')} **Animation** control

{button ,JI(`',`propCommonDialog')} **CommonDialog** control

{button ,JI(`',`propDBCombo')} **DBCombo** control

{button ,JI(`',`propDBList')} **DBList** control

{button ,JI(`',`propImageList')} **ImageList** control

{button ,JI(`',`propListView')} **ListView** control

{button ,JI(`',`propMAPIMessages')} **MAPIMessages** control

{button ,JI(`',`propMAPISession')} **MAPISession** control

{button ,JI(`',`propMasked')} **MaskedEdit** control

{button ,JI(`',`propMSChart')} **MSChart** control

{button ,JI(`',`propMSComm')} **MSComm** control

{button ,JI(`',`propMSFlexGrid')} **MSFlexGrid** control

{button ,JI(`',`propMMMCI')} **Multimedia MCI** control

{button ,JI(`',`propPicclip')} **PictureClip** control

{button ,JI(`',`propProgressBar')} **ProgressBar** control

{button ,JI(`',`propRTFBox')} **RichTextBox** control

{button ,JI(`',`propSlider')} **Slider** control

{button ,JI(`',`propSSTab')} **SSTab** control

{button ,JI(`',`propStatusBar')} **StatusBar** control

{button ,JI(`',`propTabStrip')} **TabStrip** control

{button ,JI(`',`propToolBar')} **ToolBar** control

{button ,JI(`',`propTreeView')} **TreeView** control

{button ,JI(`',`propUpDown')} **UpDown** control

{button ,JI(`',`propWinSock')} **Winsock** control

Property Pages

Animation control

AutoPlay	AutoPlay
BackStyle	BackStyle
Center	Center
OLEDropMode	OLEDropMode

Common Dialog control

CancelError	CancelError
Color	Color
Copies	Copies

DefaultExt	DefaultExt
DialogTitle	DialogTitle
FileName	FileName
Filter	Filter
FilterIndex	FilterIndex
Flags	Flags
FontName	FontName
FontSize	FontSize
FromPage	FromPage
HelpCommand	HelpCommand
HelpContext	HelpContext
HelpFile	HelpFile
HelpKey	HelpKey
InitDir	InitDir
Max	Max
MaxFileSize	MaxFileSize
Min	Min
PrinterDefault	PrinterDefault
ToPage	ToPage

DBCombo control

Appearance	Appearance
Enabled	Enabled
IntegralHeight	IntegralHeight
Locked	Locked
MatchEntry	MatchEntry
MousePointer	MousePointer
OLEDragMode	OLEDragMode
OLEDropMode	OLEDropMode
Style	Style

DBList control

Appearance	Appearance
Enabled	Enabled
IntegralHeight	IntegralHeight
Locked	Locked
MatchEntry	MatchEntry
MousePointer	MousePointer
OLEDragMode	OLEDragMode
OLEDropMode	OLEDropMode

ImageList control

Height	Height
Image	Image

Image Count	Image Count
Index	Index
Key	Key
Tag	Tag
UseMaskColor	UseMaskColor
Width	Width

ListView control

Alignment	Alignment
Appearance	Appearance
Arrange	Arrange
BorderStyle	BorderStyle
Enabled	Enabled pro
HideColumnHeaders	HideColumnHeaders
HideSelection	HideSelection
ImageList	ImageList
Index	Index
Key	Key
LabelEdit	LabelEdit
LabelWrap	LabelWrap
MousePointer	MousePointer
MultiSelect	MultiSelect
OLEDragMode	OLEDragMode
OLEDropMode	OLEDropMode
Sorted	Sorted
SortKey	SortKey
SortOrder	SortOrder
Tag	Tag
Text	Text
View	View
Width	Width

MAPIMessages control

AddressCaption	AddressCaption
AddressEditFieldCount	AddressEditFieldCount
AddressLabel	AddressLabel
AddressModifiable	AddressModifiable
AddressResolveUI	AddressResolveUI
FetchMsgType	FetchMsgType
FetchSorted	FetchSorted
FetchUnreadOnly	FetchUnreadOnly

MAPISession control

DownloadMail	DownloadMail

LogonUI	LogonUI
NewSession	NewSession
Password	Password
UserName	UserName

MaskedEdit control

AllowPrompt	AllowPrompt
AutoTab	AutoTab
BorderStyle	BorderStyle
ClipMode	ClipMode
Enabled	Enabled
Format	Format
HideSelection	HideSelection
Mask	Mask
MaxLength	MaxLength
MousePointer	MousePointer
OLEDragMode	OLEDragMode
OLEDropMode	OLEDropMode
PromptChar	PromptChar
PromptInclude	PromptInclude

MSChart control

2D	2D
3D	3D
Alignment	Alignment
Automatic scaling	Automatic scaling
Axis	Axis
Color	Color
Exclude series	Exclude series
Font style	Font style
Font	Font
Hide series	Hide series
Major divisions	Major divisions
Maximum	Maximum
Mean	Mean
Minimum	Minimum
Minor divisions	Minor divisions
Orientation	Orientation
Pattern color	Pattern color
Pattern	Pattern
Plot on 2nd Y axis	Plot on 2nd Y axis
Property Name	Property Name
Regression	Regression
Series in rows	Series in rows
Series type	Series type

Series	Series
Series	Series
Shadow	Shadow
Show legend	Show legend
Show markers	Show markers
Show markers	Show markers
Show scale	Show scale
Size	Size
Stack series	Stack series
Standard deviation	Standard deviation
Strikeout	Strikeout
Style	Style
Style	Style
Style	Style
Style	Style
Style	Style
Text	Text
Underline	Underline
Width	Width
Width	Width
Width	Width
Width	Width
Width	Width

MSComm control

CommPort	CommPort
DTREnable	DTREnable
EOFEnable	EOFEnable
Handshaking	Handshaking
InBufferSize	InBufferSize
InputLen	InputLen
NullDiscard	NullDiscard
OutBufferSize	OutBufferSize
ParityReplace	ParityReplace
Rthreshold	RThreshold
RTSEnable	RTSEnable
Settings	Settings
Sthreshold	SThreshold

MSFlexGrid control

AllowBigSelection	AllowBigSelection
AllowUserResizing	AllowUserResizing
Cols	Cols
FillStyle	FillStyle

FixedCols	FixedCols
FixedRows	FixedRows
FocusRect	FocusRect
Font	Font
Format	Format
GridLines	GridLines
GridLinesFixed	GridLinesFixed
HighLight	HighLight
MergeCells	MergeCells
MousePointer	MousePointer
PictureType	PictureType
RowHeightMin	RowHeightMin
Rows	Rows
ScrollBars	ScrollBars
SelectionMode	SelectionMode
TextStyle	TextStyle
TextStyleFixed	TextStyleFixed
WordWrap	WordWrap

Multimedia MCI control

AutoEnable	AutoEnable
BackEnabled	BackEnabled
BackVisible	BackVisible
BorderStyle	BorderStyle
DeviceType	DeviceType
EjectEnabled	EjectEnabled
EjectVisible	EjectVisible
Enabled	Enabled
FileName	FileName
Frames	Frames
MousePointer	MousePointer
NextEnabled	NextEnabled
NextVisible	NextVisible
OLEDropMode	OLEDropMode
Orientation	Orientation
PauseEnabled	PauseEnabled
PauseVisible	PauseVisible
PlayEnabled	PlayEnabled
PlayVisible	PlayVisible
PrevEnabled	PrevEnabled
PrevVisible	PrevVisible
RecordEnabled	RecordEnabled
RecordMode	RecordMode
RecordVisible	RecordVisible

Shareable	Shareable
Silent	Silent
StepEnabled	StepEnabled
StepVisible	StepVisible
StopEnabled	StopEnabled
StopVisible	StopVisible
UpdateInterval	UpdateInterval

PictureClip control

Cols	Cols
Rows	Rows

ProgressBar control

Appearance	Appearance
BorderStyle	BorderStyle
Enabled	Enabled
Max	Max
Min	Min
MousePointer	MousePointer
OLEDropMode	OLEDropMode

RichTextBox control

Appearance	Appearance
AutoVerbMenu	AutoVerbMenu
BorderStyle	BorderStyle
BulletIndent	BulletIndent
DisableNoScroll	DisableNoScroll
Enabled	Enabled
FileName [load from]	FileName [load from]
HideSelection	HideSelection
Locked	Locked
MaxLength	MaxLength
MousePointer	MousePointer
MultiLine	MultiLine
OLEDragMode	OLEDragMode
OLEDropMode	OLEDropMode
RightMargin	RightMargin
ScrollBars	ScrollBars

Slider control

Enabled	Enabled
LargeChange	LargeChange
Max	Max
Min	Min

MousePointer	MousePointer
OLEDropMode	OLEDropMode
Orientation	Orientation
SelectRange	SelectRange
SelLength	SelLength
SelStart	SelStart
SmallChange	SmallChange
TickFrequency	TickFrequency
TickStyle	TickStyle

SSTab control

Current Tab	CurrentTab
Enabled	Enabled
MousePointer	MousePointer
OLEDropMode	OLEDropMode
Orientation	Orientation
ShowFocusRect	ShowFocusRect
Style	Style
Tab Count	Tab Count
TabCaption	TabCaption
TabHeight	TabHeight
TabMaxWidth	TabMaxWidth
TabsPerRow	TabsPerRow
WordWrap	WordWrap

StatusBar control

Actual Width	Actual Width
Alignment	Alignment
AutoSize	AutoSize
Bevel	Bevel
Enabled	Enabled
Index	Index
Key	Key
Minimum Width	Minimum Width
MousePointer	MousePointer
OLEDropMode	OLEDropMode
Picture	Picture
ShowTips	ShowTips
SimpleText	SimpleText
Style	Style
Tag	Tag
Text	Text
ToolTipText	ToolTipText

TabStrip control

Caption	Caption
Enabled	Enabled
Image	Image
ImageList	ImageList
Index	Index
Key	Key
MousePointer	MousePointer
MultiRow	MultiRow
OLEDropMode	OLEDropMode
ShowTips	ShowTips
Style	Style
TabFixedHeight	TabFixedHeight
TabFixedWidth	TabFixedWidth
TabWidthStyle	TabWidthStyle
Tag	Tag
ToolTipText	ToolTipText

TreeView control

Appearance	Appearance
BorderStyle	BorderStyle
Enabled	Enabled
HideSelection	HideSelection
ImageList	ImageList
Indentation	Indentation
LabelEdit	LabelEdit
LineStyle	LineStyle
MousePointer	MousePointer
OLEDragMode	OLEDragMode
OLEDropMode	OLEDropMode
PathSeparator	PathSeparator
Sorted	Sorted
Style	Style

ToolBar control

AllowCustomize	AllowCustomize
Appearance	Appearance
BorderStyle	BorderStyle
ButtonHeight	ButtonHeight
ButtonWidth	ButtonWidth
Caption	Caption
Description	Description
Enabled	Enabled
HelpContextID	HelpContextID

HelpFile	HelpFile
Image	Image
ImageList	ImageList
Index	Index
Key	Key
MixedState	MixedState
MousePointer	MousePointer
OLEDropMode	OLEDropMode
ShowTips	ShowTips
Style	Style
Tag	Tag
ToolTipText	ToolTipText
Value	Value
Visible	Visible
Width	Width
Wrappable	Wrappable

UpDown control

Alignment	Alignment
AutoBuddy	AutoBuddy
BuddyControl	BuddyControl
BuddyProperty	BuddyProperty
Increment	Increment
Max	Max
Min	Min
OLEDropMode	OLEDropMode
Orientation	Orientation
SyncBuddy	SyncBuddy
Value	Value
Wrap	Wrap

Winsock control

LocalPort	LocalPort
Protocol	Protocol
RemoteHost	RemoteHost
RemotePort	RemotePort

RightToLeft Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproRightToLeftPropertyActiveXControlsC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRightToLeftPropertyActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRightToLeftPropertyActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRightToLeftPropertyActiveXControlsS"}

Returns a boolean value that indicates the text display direction and controls the visual appearance on a bidirectional system.

Syntax

object.**RightToLeft**

The **RightToLeft** property syntax has this part:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.

Settings

The possible boolean return values from the **RightToLeft** property are:

Setting	Description
True	The control is running on a bi-directional platform, such as Arabic Windows95 or Hebrew Windows95, and text is running from right to left. The control should modify its behavior, such as putting vertical scroll bars at the left side of a text or list box, putting labels to the right of text boxes, etc.
False	The control should act as though it was running on a non-bidirectional platform, such as English Windows95, and text is running from left to right. If the container does not implement this ambient property, this will be the default value.

Refresh Method (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRefreshMethodActiveXControlsC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRefreshMethodActiveXControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRefreshMethodActiveXControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRefreshMethodActiveXControlsS"}

Forces a complete repaint of a form or control.

Syntax

object.**Refresh**

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

Remarks

Use the **Refresh** method when you want to:

- Completely display one form while another form loads.
- Update the contents of a file-system list box, such as a **FileListBox** control.
- Update the data structures of a **Data** control.

Refresh can't be used on MDI forms , but can be used on MDI child forms. You can't use **Refresh** on **Menu** or **Timer** controls.

Generally, painting a form or control is handled automatically while no events are occurring. However, there may be situations where you want the form or control updated immediately. For example, if you use a file list box, a directory list box, or a drive list box to show the current status of the directory structure, you can use **Refresh** to update the list whenever a change is made to the directory structure.

You can use the **Refresh** method on a **Data** control to open or reopen the database (if the **DatabaseName**, **ReadOnly**, **Exclusive**, or **Connect** property settings have changed) and rebuild the dynaset in the control's **Recordset** property.