Change Event (ToolBar, Slider Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtChangeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtChange;vbevtChangeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtChangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtChangeOS"}

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

Syntax

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

The Change event syntax has these parts:

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

Remarks

- Slider — generated when the **Value** property changes, either through code, or when the user moves the control's slider.
- Toolbar — generated after the end user customizes a Toolbar control's toolbar using the Customize Toolbar dialog box.

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

Note A Change event procedure can sometimes cause a cascading event. This occurs when the control's Change event alters the control's contents 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.

Trappable Errors for the Windows Common Controls

{ewc HLP95EN.DLL,DYNALINK,"See Also":"msgWindows95ControlsErrorsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"msgWindows95ControlsErrorsX":1}

The following tables list the trappable errors and constants for the Windows Common Controls in the ComCtl32.OCX and ComCt232.OCX files.

The following are the errors for the Windows Common Controls located in the ComCtl32.OCX file. Controls included in this file are: **TabStrip**, **Toolbar**, **StatusBar**, **ProgressBar**, **TreeView**, **ListView**, **ImageList**, and **Slider**.

Constant	Value	Description
ccInvalidProcedureCAll	5	Invalid procedure call
ccOutofMemory	7	Out of memory The operation could not allocate enough memory.
ccTypeMismatch	13	Type Mismatch One of the arguments could not be converted to the correct data type.
ccInvalidPropertyValue	380	Invalid property value A value has been assigned to a property, that is outside its permissible range.
ccSetNotSupportedAtRuntime	382	Property cannot be set at run time
ccSetNotSupported	383	Property is read-only
ccSetNotPermitted	387	Property can't be set on this control
ccGetNotSupported	394	Property is write-only
ccInvalidClipboardFormat	460	Invalid clipboard format
ccInvalidObjectUse	425	Invalid object use
ccDataObjectLocked	672	DataObject formats list may not be cleared or expanded outside of the OLEStartDrag event
ccExpectedAnArgument	673	Expected at least one argument
ccRecursiveOleDrag	674	Illegal recursive invocation of OLE drag and drop
ccIndexOutOfBounds	35600	Index out of bounds
ccElemNotFound	35601	Element not found
ccNonUniqueKey	35602	Key is not unique in collection
ccInvalidKey	35603	Invalid key
ccCol1MustBeLeftAligned	35604	When the ListView control's **View** property is set to 3 (Report), the left-most column (column 1) can only be left aligned. Any attempt to set the alignment to another value will result in this error.

ccElemNotPartOfCollection	35605	This item's control has been deleted
ccCollectionChangedDuringEnum	35606	Control's collection has been modified
ccMissingRequiredArg	35607	Required argument is missing
ccBadObjectReference	35610	Invalid object
ccReadOnlyIfHasImages	35611	Property is read-only if image list contains images
ccImageListMustBeInitialized	35613	ImageList must be initialized before it can be used
ccWouldIntroduceCycle	35614	This would introduce a cycle
ccNotSameSize	35615	All images in list must be same size
ccMaxPanelsExceeded	35616	Maximum Panels Exceeded
ccImageListLocked	35617	ImageList cannot be modified while another control is bound to it
ccMaxButtonsExceeded	35619	Maximum Buttons Exceeded
ccCircularReference	35700	Circular object referencing is not allowed

The following are the errors for the Windows Common Controls located in the ComCt232.OCX file. Controls included in this file are: **UpDown** and **Animation**.

Constant	Value	Description
cc2InvalidProcedureCall	5	Invalid procedure call
cc2BadFileNameOrNumber	52	Bad file name or number
cc2FileNotFound	53	File not found
cc2InvalidPropertyValue	380	Invalid property value
cc2SetNotSupportedAtRuntime	382	Property cannot be set at runtime
cc2SetNotSupported	383	Property is read-only
cc2InvalidObjectUse	425	Invalid object use
cc2InvalidClipboardFormat	460	Invalid clipboard format
cc2DataObjectLocked	672	DataObject formats list may not be cleared or expanded outside of the OLEStartDrag event
cc2ExpectedAnArgument	673	Expected at least one argument.
cc2InconsistentObject	35750	Internal state of the control has become corrupted
cc2ErrorDuringSet	35751	Unable to set property
cc2ErrorOpeningVideo	35752	Unable to open AVI file
cc2ErrorPlayingVideo	35753	Unable to play AVI file
cc2NoValidBuddyCtl	35754	BuddyControl property must be set first
cc2VideoNotOpen	35755	Must open AVI file first
cc2AutoBuddyNotSet	35756	AutoBuddy not set, no potential buddy controls found

cc2ErrorStoppingVideo	35757	Error trying to stop playing video file
cc2ErrorClosingVideo	35758	Error closing open video file
cc2CantStopAutoPlay	35759	Stop method does not effect AutoPlay property
cc2BuddyNotASibling	35760	BuddyControl must be a separate control within the same container
cc2NoUpDownAsBuddy	35761	An UpDown control cannot be buddied with another UpDown control.

Error Messages (RichTextBox Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmsgErrorMessagesRichTextBoxC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmsgErrorMessagesRichTextBoxS"}

The following are the errors for the **RichTextBox** control, located in the RichTx32.OCX file:

Constant	Value	Description
rtfInvalidProcedureCall	5	Invalid procedure call
rtfOutOfMemory	7	Out of memory
rtfPathFileAccessError	75	The specified path/file name cannot be accessed or is invalid
rtfInvalidFileFormat	321	Invalid file format
rtfInvalidPropertyValue	380	Invalid property value
rtfInvalidPropertyArrayIndex	381	Invalid property array index
rtfSetNotSupported	383	Property is read-only
rtfSetNotPermitted	387	Property cannot be set
rtfGetNotSupported	394	Property is write-only
rtfInvalidObjectUse	425	Invalid object use
rtfDataObjectLocked	672	DataObject formats list may not be cleared or expanded outside of the OLEStartDrag event
rtfExpectedAnArgument	673	Expected at least one argument
rtfInvalidCharPosition	32000	Invalid character position
rtfInvalidHdc	32001	Invalid HDC
rtfCannotLoadFile	32002	Unable to load specified file
rtfInvalidKeyName	32005	Invalid or missing key name
rtfInvalidClassName	32006	Invalid or missing OLE class name
rtfKeyNotFound	32007	Key does not exist
rtfOLESourceRquired	32008	Required source document or class name is missing
rtfNonUniqueKey	32009	Key is not unique in collection
rtfInvalidObject	32010	Invalid object
rtfProtected	32011	The operation cannot be performed on protected text
rtfOleCreate	32012	Unable to create object
rtfOleServer	32013	Unable to start server application

ImageList Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjImageListC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjImageListX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjImageListP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjImageListM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjImageListE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjImageListS"}

An **ImageList** control contains a collection of **ListImage** objects, each of which can be referred to by its index or key. The **ImageList** control is not meant to be used alone, but as a central repository to conveniently supply other controls with images.

Syntax

ImageList

Remarks

You can use the **ImageList** control with any control that assigns a **Picture** object to a **Picture** property. For example, the following code assigns the first **ListImage** object in a **ListImages** collection to the **Picture** property of a newly created **StatusBar** panel:

```
Dim pnlX As Panel
Set pnlX = StatusBar1.Panels.Add() ' Add a new Panel object.
Set pnlX.Picture = ImageList1.ListImages(1).Picture ' Set Picture.
```

Note You must use the **Set** statement when assigning an image to a **Picture** object.

Images of different sizes can be added to an **ImageList** control, but it constrains them all to be the same size. The size of the **ListImage** objects is determined by one of the following:

- The setting of **ImageWidth** and **ImageHeight** properties before any images are added.
- The dimensions of the first image added.

You are not limited to any particular image size, but the total number of images that can be loaded is limited by the amount of available memory.

At design time, you can add images using the General tab of the ImageList Control Properties dialog box. At run time, you can add images using the **Add** method for the **ListImages** collection.

Besides storing **Picture** objects, the **ImageList** control can also perform graphical operations on images before assigning them to other controls. For example, the **Overlay** method creates a composite image from two different images.

Additionally, you can bind one or more **ImageList** controls to certain other Windows 95 common controls to conserve system resources. These include the **ListView**, **ToolBar**, **TabStrip**, and **TreeView** controls. In order to use an **ImageList** with one of these controls, you must associate a particular **ImageList** with the control through an appropriate property. For the **ListView** control, you must set the **Icons** and **SmallIcons** properties to **ImageList** controls. For the **TreeView**, **TabStrip**, and **Toolbar** controls, you must set the **ImageList** property to an **ImageList** control.

For these controls, you can specify an **ImageList** at design time using the Custom Properties dialog box. At run time, you can also specify an **ImageList** which sets a **TreeView** control's **ImageList** property, as in the following example:

```
TreeView1.ImageList = ImageList1  ' Specify ImageList
```

Once you associate an **ImageList** with a control, you can use the value of either the **Index** or **Key** property to refer to a **ListImage** object in a procedure. The following example sets the **Image** property of a **TreeView** control's third **Node** object to the first **ListImage** object in an **ImageList** control:

```
' Use the value of the Index property of ImageList1.
TreeView1.Nodes(3).Image = 1
```

```
' Or use the value of the Key property.
TreeView1.Nodes(3).Image = "image 1"    ' Assuming Key is "image 1."
```

Distribution Note The **ImageList** control is part of a group of ActiveX controls that are found in the COMCTL32.OCX file. To use the **ImageList** control in your application, you must add the COMCTL32.OCX file to the project. When distributing your application, install the COMCTL32.OCX file in the user's Microsoft Windows System or System32 directory. For more information on how to add an ActiveX control to a project, see the *Programmer's Guide*.

ListImage Object, ListImages Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjListImageC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjListImageX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vamthItem;vbobjListImageP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjListImageM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjListImageE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjListImageS"}

- A **ListImage** object is a bitmap of any size that can be used in other controls.
- A **ListImages** collection is a collection of **ListImage** objects.

Syntax

imagelist.**ListImages**
imagelist.**ListImages(***index***)**

The syntax lines above refer to the collection and to individual elements in the collection, respectively, according to standard collection syntax.

The **ListImage** Object, **ListImages** Collection syntaxes have these parts:

Part	Description
imagelist	Required. An object expression that evaluates to an object in the Applies To list.
index	An integer or string that uniquely identifies the object in the collection. The integer is the value of the **Index** property; the string is the value of the **Key** property.

Remarks

The **ListImages** collection is a 1-based collection.

You can add and remove a **ListImage** at design time using the General tab of the ImageList Control Properties page, or at run time using the **Add** method for **ListImage** objects.

Each item in the collection can be accessed by its index or unique key. For example, to get a reference to the third **ListImage** object in a collection, use the following syntax:

```
Dim imgX As ListImage
   ' Reference by index number.
Set imgX = ImageList.ListImages(3)
   ' Or reference by unique key.
Set imgX = ImageList1.ListImages("third") ' Assuming Key is "third."
   ' Or use Item method.
Set imgX = ImageList1.ListImages.Item(3)
```

Each **ListImage** object has a corresponding mask that is generated automatically using the **MaskColor** property. This mask is not used directly, but is applied to the original bitmap in graphical operations such as the **Overlay** and **Draw** methods.

Add Method (ListImages Collection)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthAddImagesC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAddListImagesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAddListImagesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAddListImagesS"}

Adds a **ListImage** object to a **ListImages** collection. Doesn't support named arguments.

Syntax

object.**Add(***index*, *key*, *picture***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
index	Optional. An integer specifying the position where you want to insert the **ListImage**. If no *index* is specified, the **ListImage** is added to the end of the **ListImages** collection.
key	Optional. A unique string that identifies the **ListImage** object. Use this value to retrieve a specific **ListImage** object. An error occurs if the key is not unique.
picture	Required. Specifies the picture to be added to the collection.

Remarks

The **ListImages** collection is a 1-based collection.

You can load either bitmaps or icons into a **ListImage** object. To load a bitmap or icon, you can use the **LoadPicture** function, as follows:

```
Set imgX = ImageList1.ListImages.Add(,,LoadPicture("file name"))
```

You can also load a **Picture** object directly into the **ListImage** object. For example, this example loads a **PictureBox** control's picture into the **ListImage** object:

```
Set imgX = ImageList1.ListImages.Add(,,Picture1.Picture)
```

If no **ListImage** objects have been added to a **ListImages** collection, you can set the **ImageHeight** and **ImageWidth** properties before adding the first **ListImage** object. The first **ListImage** object you add to a collection can be any size. However, all subsequent **ListImage** objects will be forced to be the same size as the first **ListImage** object. Once a **ListImage** object has been added to the collection, the **ImageHeight** and **ImageWidth** properties become read-only properties, and any image added to the collection must have the same **ImageHeight** and **ImageWidth** values.

You should use the **Key** property to reference a **ListImage** object if you expect the value of the **Index** property to change. For example, if you allow users to add and delete their own images to the collection, the value of the **Index** property may change.

When a **ListImage** object is added to the collection, a reference to the newly created object is returned. You can use the reference to set other properties of the **ListImage**, as follows:

```
Dim imgX As ListImage
Dim I As Integer
   Set imgX = ImageList1.ListImages. _
   Add(,,LoadPicture("icons\comm\net01.ico"))
   imgX.Key = "net connect" ' Use the new reference to assign Key.
```

Add Method (ListImages Collection) Example

This example adds several images to a **ListImages** collection, and then uses the images in a **TreeView** control. To try the example, place **ImageList** and **TreeView** controls on a form, and paste the code into the form's Declarations section. Run the example to see the **TreeView** populated with pictures from the **ImageList**.

```
Private Sub Form_Load()
   Dim imgX As ListImage
   ' Load three icons into the ImageList control's collection.
   Set imgX = ImageList1.ListImages. _
   Add(,"rocket", LoadPicture("icons\industry\rocket.ico"))
   Set imgX = ImageList1.ListImages. _
   Add(,"plane",LoadPicture("icons\industry\plane.ico"))
   Set imgX = ImageList1.ListImages. _
   Add(,"car",LoadPicture("icons\industry\cars.ico"))

   ' Set TreeView control's ImageList property.
   Set TreeView1.ImageList = ImageList1

   ' Create a Treeview, and use ListImage objects for its images.
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,,"Rocket")
   nodX.Image = 1 ' Use the Index property of image 1.
   Set nodX = TreeView1.Nodes.Add(,,,"Plane")
   nodX.Image = "plane" ' Use the Key property of image 2.
   Set nodX = TreeView1.Nodes.Add(,,,"Car")
   nodX.Image = "car"   ' Use the Key property of image 3.
End Sub
```

Draw Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthDrawC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthDrawX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthDrawA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthDrawS"}

Draws an image into a destination device context, such as a **PictureBox** control, after performing a graphical operation on the image. Doesn't support named arguments.

Syntax

object.**Draw (***hDC*, *x*,*y*, *style***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
hDC	Required. A value set to the target object's **hDC** property.
x,y	Optional. The coordinates used to specify the location within the device context where the image will be drawn. If you don't specify these, the image is drawn at the origin of the device context.
style	Optional. Specifies the operation performed on the image, as described in Settings.

Settings

The settings for *style* are:

Constant	Value	Description
imlNormal	0	(Default) Normal. Draws the image with no change.
imlTransparent	1	Transparent. Draws the image using the **MaskColor** property to determine which color of the image will be transparent.
imlSelected	2	Selected. Draws the image dithered with the system highlight color.
imlFocus	3	Focus. Draws the image dithered and striped with the highlight color creating a hatched effect to indicate the image has the focus.

Remarks

The **hDC** property is a handle (a number) that the Windows operating system uses for internal reference to an object. You can paint in the internal area of any control that has an **hDC** property. In Visual Basic, these include the **Form** object, **PictureBox** control, and **Printer** object.

Because an object's **hDC** can change while an application is running, it is better to specify the **hDC** property rather than an actual value. For example, the following code ensures that the correct **hDC** value is always supplied to the **ImageList** control:

```
ImageList1.ListImages(1).Draw Form1.hDC
```

Draw Method Example

This example loads an image into an **ImageList** control. When you click the form, the image is drawn on the form in four different styles. To try the example, place an **ImageList** control on a form and paste the code into the form's Declarations section. Run the example and click the form.

```
Private Sub Form_Load()
   Dim X As ListImage
   'Load one image into the ImageList.
   Set X = ImageList1.ListImages. _
   Add(, , LoadPicture("bitmaps\assorted\intl_no.bmp"))
End Sub

Private Sub Form_Click()
   Dim space, intW As Integer      ' Create spacing variables.

   ' Use the ImageWidth property for spacing.
   intW = ImageList1.ImageWidth
   space = Form1.Font.Size * 2 ' Use the Font.Size for height spacing.

   ScaleMode = vbPoints ' Set ScaleMode to points.
   Cls ' Clear the form.

   ' Draw the image with Normal style.
   ImageList1.ListImages(1).Draw Form1.hDC, , space,imlNormal
   ' Set MaskColor to red, which will become transparent.
   ImageList1.MaskColor = vbRed
   ' Draw the image with red (MaskColor) the transparent color.
   ImageList1.ListImages(1).Draw Form1.hDC, intW, space,imlTransparent
   ' Draw image with the Selected style.
   ImageList1.ListImages(1).Draw Form1.hDC, intW * 2,space,imlSelected
   ' Draw image with Focus style.
   ImageList1.ListImages(1).Draw Form1.hDC, intW * 3, space,imlFocus

   ' Print a caption for the images.
   Print _
   "Normal              Transparent           Selected         Focus"

End Sub
```

ExtractIcon Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthExtractIconC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthExtractIconX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthExtractIconA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthExtractIconS"}

Creates an icon from a bitmap in a **ListImage** object of an **ImageList** control and returns a reference to the newly created icon.

Syntax

object.**ExtractIcon**

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

Remarks

You can use the icon created with the **ExtractIcon** method like any other icon. For example, you can use it as a setting for the **MouseIcon** property, as the following code illustrates:

```
Set Command1.MouseIcon = ImageList1.ListImages(1).ExtractIcon
```

ExtractIcon Method Example

This example loads a bitmap into an **ImageList** control. When the user clicks the form, the **ExtractIcon** method is used to create an icon from the bitmap, and that icon is used as a setting in the **Form** object's **MouseIcon** property. To try the example, place an **ImageList** control on a form and paste the code into the form's Declarations section. Run the example and click the form.

```
Private Sub Form_Load()
   Dim imgX As ListImage
   Set imgX = ImageList1.ListImages. _
   Add(, , LoadPicture("bitmaps\assorted\balloon.bmp"))
End Sub

Private Sub Form_Click()
   Dim picX As Picture
   Set picX = ImageList1.ListImages(1).ExtractIcon  ' Make an icon.

   With Form1
   .MouseIcon =  picX   ' Set new icon.
   .MousePointer = vbCustom  ' Set to custom icon.
   End With
End Sub
```

ImageHeight, ImageWidth Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproImageHeightC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproImageHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproImageHeightA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproImageHeightS"}

- The **ImageHeight** property returns or sets the height of **ListImage** objects in an **ImageList** control.
- The **ImageWidth** property returns or sets the width of **ListImage** objects in an **ImageList** control.

Syntax

object.**ImageHeight**
object.**ImageWidth**

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

Remarks

Both height and width are measured in pixels. All images in a **ListImages** collection have the same height and width properties.

When an **ImageList** contains no **ListImage** objects, you can set both **ImageHeight** and **ImageWidth** properties. However, once a **ListImage** object has been added, all subsequent images must be of the same height and width as the first object. If you try to add an image of a different size, an error is returned.

ImageHeight, ImageWidth Properties Example

This example loads an icon into an **ImageList** control, and uses the image in a **ListView** control. When the user clicks the form, the code uses the **ImageHeight** property to adjust the height of the **ListView** control to accommodate the **ListImage** object. To try the example, place **ImageList** and **ListView** controls on a form and paste the code into the form's Declarations section. Run the example and click the form.

```
Private Sub Form_Load()
   ' Create variables for ImageList and ListView objects.
   Dim imgX As ListImage
   Dim itmX As ListItem

   Form1.ScaleMode = vbPixels ' Make sure ScaleMode is set to pixels.

   ListView1.BorderStyle = FixedSingle ' Show border.
   ' Shorten ListView control so later contrast is more obvious.
   ListView1.Height = 50

   ' Put a large bitmap into the ImageList.
   Set imgX = ImageList1.ListImages. _
   Add(,, LoadPicture("bitmaps\gauge\vert.bmp"))

   ListView1.Icons = ImageList1 ' Set Icons property.

   ' Add an item to the ListView control.
   Set itmX = ListView1.ListItems.Add()
   itmX.Icon = 1  ' Set Icon property to ListImage 1 of ImageList.
   itmX.Text = "Thermometer" ' Set text of ListView ListItem object.
End Sub

Private Sub Form_Click()
   Dim strHW As String

   strHW = "Height: " & ImageList1.ImageHeight & _
   "   Width: " & ImageList1.ImageWidth
   caption = strHW   ' Show dimensions.
   ' Enlarge ListView to accommodate the tallest image.
   ListView1.Height = ImageList1.ImageHeight + 50
End Sub
```

ListImages Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproListImagesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproListImagesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproListImagesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproListImagesS"}

Returns a reference to a collection of **ListImage** objects in an **ImageList** control.

Syntax

object.**ListImages**

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

Remarks

You can manipulate **ListImage** objects using standard collection methods (for example, the **Add** and **Clear** methods). Each member of the collection can be accessed by its index or unique key. These are stored in the **Index** and **Key** properties, respectively, when **ListImage** is added to a collection.

ListImages Property Example

This example adds three **ListImage** objects to a **ListImages** collection and uses them in a **ListView** control. The code refers to the **ListImage** objects using both their **Key** and **Item** properties. To try the example, place **ImageList** and **ListView** controls on a form and paste the code into the form's Declarations section. Run the example.

```
Private Sub Form_Load()
   Dim imgX As ListImage
   ' Add images to ListImages collection.
   Set imgX = ImageList1. _
   ListImages.Add(,"rocket",LoadPicture("icons\industry\rocket.ico"))
   Set imgX = ImageList1. _
   ListImages.Add(,"jet",LoadPicture("icons\industry\plane.ico"))
   Set imgX = ImageList1. _
   ListImages.Add(,"car",LoadPicture("icons\industry\cars.ico"))

   ListView1.Icons = ImageList1 ' Set Icons property.

   ' Add Item objects to the ListView control.
   Dim itmX as ListItem
   Set itmX = ListView1.ListItems.Add()
   ' Reference by index.
   itmX.Icon = 1
   itmX.Text = "Rocket" ' Set Text string.
   Set itmX = ListView1.ListItems.Add()
   ' Reference by key ("jet").
   itmX.Icon = "jet"
   itmX.Text = "Jet" ' Set Text string.
   Set itmX = ListView1.ListItems.Add()
   itmX.Icon = "car"
   itmX.Text = "Car" ' Set Text string.
End Sub
```

MaskColor Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vafctQBColor;vafctRGB;vbproBooksOnlineJumpTopic;vbproMaskColorC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMaskColorX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMaskColorA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMaskColorS"}

Returns or sets the color used to create masks for an **ImageList** control.

Syntax

object.**MaskColor** [= *color*]

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
color	A value or constant that determines the color used to create masks. You can specify colors using either Visual Basic intrinsic constants, the **QBColor** function, or the **RGB** function.

Remarks

Every image in a **ListImages** collection has a corresponding mask associated with it. The mask is a monochrome image derived from the image itself, automatically generated using the **MaskColor** property as the specific color of the mask. This mask is not used directly, but is applied to the original bitmap in graphical operations such as the **Overlay** and **Draw** methods. For example, the **MaskColor** property determines which color of an image will be transparent in the **Overlay** method.

MaskColor Property Example

This example loads several bitmaps into an **ImageList** control. As you click the form, one **ListImage** object is overlaid on one of the other **ListImage** objects. To try the example, place an **ImageList** control and a **Picture** control on a form and paste the code into the form's Declarations section. Run the program and click the form.

```
Private Sub Form_Load()
   Dim imgX As ListImage

   ' Load bitmaps.
   Set imgX = ImageList1.ListImages. _
   Add(, "No", LoadPicture("bitmaps\assorted\Intl_No.bmp"))
   Set imgX = ImageList1.ListImages. _
   Add(, , LoadPicture("bitmaps\assorted\smokes.bmp"))
   Set imgX = ImageList1.ListImages. _
   Add(, , LoadPicture("bitmaps\assorted\beany.bmp"))

   ScaleMode = vbPixels
   ' Set MaskColor property.
   ImageList1.MaskColor = vbGreen
   ' Set the form's BackColor to white.
   Form1.BackColor = vbWhite
End Sub

Private Sub Form_Click()
   Static intCount As Integer ' Static variable to count images.

   ' Reset variable to 2 if it is over the ListImages.Count value.
   If intCount > ImageList1.ListImages.Count Or intCount < 1 Then
      intCount = 2 ' Reset to second image
   End If

   ' Overlay ListImage(1) over ListImages 2-3.
   Picture1.Picture = ImageList1.Overlay(intCount, 1)
   ' Increment count.
   intCount = intCount + 1

   ' Create variable to hold ImageList.ImageWidth value.
   Dim intW
   intW = ImageList1.ImageWidth

   ' Draw images onto the form for reference. Use the ImageWidth
   ' value to space the images.
   ImageList1.ListImages(1).Draw Form1.hDC, 0, 0, imlNormal
   ImageList1.ListImages(2).Draw Form1.hDC, 0, intW, imlNormal
   ImageList1.ListImages(3).Draw Form1.hDC, 0, intW * 2, imlNormal
End Sub
```

UseMaskColor Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproUseMaskColorPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproUseMaskColorPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproUseMaskColorPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproUseMaskColorPropertyS"}

Returns or sets a value which determines whether the **ImageList** control will use the **MaskColor** property.

Syntax

object.**UseMaskColor** [= *boolean*]

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying whether the color specified in the **MaskColor** property is to be used for the control.

Overlay Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthOverlayC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthOverlayX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthOverlayA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthOverlayS"}

Draws one image from a **ListImages** collection over another, and returns the result. Doesn't support named arguments.

Syntax

object.**Overlay (***index1*, *index2***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
index1	Required. An integer (**Index** property) or unique string (**Key** property) that specifies the image to be overlaid.
index2	Required. An integer (**Index** property) or unique string (**Key** property) that specifies the image to be drawn over the object specified in *index1*. The color of the image that matches the **MaskColor** property is made transparent. If no color matches, the image is drawn opaquely over the other image.

Remarks

Use the **Overlay** method in conjunction with the **MaskColor** property to create a single image from two disparate images. The **Overlay** method imposes one bitmap over another to create a third, composite image. The **MaskColor** property determines which color of the overlaying image is transparent.

The *index* can be either an index or a key. For example, to overlay the first picture in the collection with the second:

```
Set Picture1.Picture = ImageList1.Overlay(1,2) ' Reference by Index.
   'Or reference by Key property.
Set Picture1.Picture = ImageList1.Overlay("First", "Second")
```

Overlay Method Example

This example loads five **ListImage** objects into an **ImageList** control and displays any two images in two **PictureBox** controls. For each **PictureBox**, select an image to display from one of the two **ComboBox** controls. When you click the form, the code uses the **Overlay** method to create a third image that is displayed in a third **PictureBox** control. To try the example, place an **ImageList** control, two **ComboBox** controls, and three **PictureBox** controls on a form and paste the code into the form's Declarations section. Run the example and click the form.

```
Private Sub Form_Load()
   Dim X As ListImage
   ' Add 5 images to a ListImages collection.
   Set X = ImageList1.ListImages. _
    Add(, , LoadPicture("icons\elements\moon05.ico"))
   Set X = ImageList1.ListImages. _
    Add(, , LoadPicture("icons\elements\snow.ico"))
   Set X = ImageList1.ListImages. _
    Add(, , LoadPicture("icons\writing\erase02.ico"))
   Set X = ImageList1.ListImages. _
    Add(, , LoadPicture("icons\writing\note06.ico"))
   Set X = ImageList1.ListImages. _
    Add(, , LoadPicture("icons\flags\flgfran.ico"))

   With combo1 ' Populate the first ComboBox.
      .AddItem "Moon"
      .AddItem "Snowflake"
      .AddItem "Pencil"
      .AddItem "Note"
      .AddItem "Flag"
      .ListIndex = 0
   End With

   With combo2 ' Populate the second ComboBox.
      .AddItem "Moon"
      .AddItem "Snowflake"
      .AddItem "Pencil"
      .AddItem "Note"
      .AddItem "Flag"
      .ListIndex = 2
   End With

   Picture1.BackColor = vbWhite    ' Make BackColor white.
   Picture2.BackColor = vbWhite
   Picture3.BackColor = vbWhite
End Sub

Private Sub Form_Click()
   ' Overlay the two images, and display in PictureBox3.
   Set Picture3.Picture = ImageList1. _
    Overlay(combo1.ListIndex + 1, combo2.ListIndex + 1)
End Sub

Private Sub combo1_Click()
   ' Change PictureBox to reflect ComboBox selection.
   Set Picture1.Picture = ImageList1. _
    ListImages(combo1.ListIndex + 1).ExtractIcon
```

```
End Sub

Private Sub combo2_Click()
   ' Change PictureBox to reflect ComboBox selection.
   Set Picture2.Picture = ImageList1. _
    ListImages(combo2.ListIndex + 1).ExtractIcon
End Sub
```

ImageList Control Constants

Constant	Value	Description
imlNormal	0	Image is drawn with no change.
imlTransparent	1	Image is drawn transparently.
imlSelected	2	Image is drawn selected.
imlFocus	3	Image is drawn with focus.

hImageList Property (ImageList Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbprohImageListPropertyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbprohImageListPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbprohImageListPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbprohImageListPropertyS"}

Returns a handle to an **ImageList** control.

Syntax

object.**hImageList**

The *object* placeholder represents an object expression that evaluates to an **ImageList** control.

Remarks

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

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

ListView Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjListViewC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjListViewX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjListviewP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjListViewM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjListViewE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjListViewS"}

The **ListView** control displays items using one of four different views. You can arrange items into columns with or without column headings as well as display accompanying icons and text.

Syntax

ListView

Remarks

With a **ListView** control, you can organize list entries, called **ListItem** objects, into one of four different views:

- Large (standard) Icons
- Small Icons
- List
- Report

The **View** property determines which view the control uses to display the items in the list. You can also control whether the labels associated with items in the list wrap to more than one line using the **LabelWrap** property. In addition, you can manage how items in the list are sorted and how selected items appear.

The **ListView** control contains **ListItem** and **ColumnHeader** objects. A **ListItem** object defines the various characteristics of items in the **ListView** control, such as:

- A brief description of the item.
- Icons that may appear with the item, supplied by an **ImageList** control.
- Additional pieces of text, called subitems, associated with a **ListItem** object that you can display in Report view.

You can choose to display column headings in the **ListView** control using the **HideColumnHeaders** property. They can be added at both design and run time. At design time, you can use the Column Headers tab of the **ListView** Control Properties dialog box. At run time, use the **Add** method to add a **ColumnHeader** object to the **ColumnHeaders** collection.

Distribution Note The **ListView** control is part of a group of ActiveX controls that are found in the COMCTL32.OCX file. To use the **ListView** control in your application, you must add the COMCTL32.OCX file to the project. When distributing your application, install the COMCTL32.OCX file in the user's Microsoft Windows System or System32 directory. For more information on how to add an ActiveX control to a Visual Basic project, see the Visual Basic *Programmer's Guide*.

ColumnHeader Object, ColumnHeaders Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjColumnHeaderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjColumnHeaderX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vamthItem;vbobjColumnHeaderP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vamthClear;vamthRemove;vbobjColumnHeaderM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjColumnHeaderE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjColumnHeaderS"}

- A **ColumnHeader** object is an item in a **ListView** control that contains heading text.
- A **ColumnHeaders** collection contains one or more **ColumnHeader** objects.

Syntax

listview.**ColumnHeaders**
listview.**ColumnHeaders(***index***)**

The syntax lines above refer to the collection and to individual elements in the collection, respectively, according to the standard collection syntax.

The **ColumnHeader** object, **ColumnHeaders** collection syntax has these parts:

Part	Description
listview	An object expression that evaluates to a **ListView** control.
index	Either an integer or string that uniquely identifies a member of an object collection. An integer would be the value of the **Index** property; a string would be the value of the **Key** property.

Remarks

You can view **ColumnHeader** objects in Report view only.

You can add **ColumnHeader** objects to a **ListView** control at both design time and run time.

With a **ColumnHeader** object, a user can:

- Click it to trigger the **ColumnClick** event and sort the items based on that data item.
- Grab the object's right border and drag it to adjust the width of the column.
- Hide **ColumnHeader** objects in Report view.

There is always one column in the **ListView** control, which is Column 1. This column contains the actual **ListItem** objects; not their subitems. The second column (Column 2) contains subitems. Therefore, you always have one more **ColumnHeader** object than subitems and the **ListItem** object's **SubItems** property is a 1-based array of size `ColumnHeaders.Count - 1`.

The number of **ColumnHeader** objects determines the number of subitems each **ListItem** object in the control can have. When you delete a **ColumnHeader** object, all of the subitems associated with the column are also deleted, and each **ListItem** object's subitem array shifts to update the indices of the **ColumnHeader**, causing the remaining column headers' **SubItemIndex** properties to change.

Add Method (ColumnHeaders Collection)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthAddColumnHeadersC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAddColumnHeadersX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAddColumnHeadersA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAddColumnHeadersS"}

Adds a **ColumnHeader** object to a **ColumnHeaders** collection in a **ListView** control. Doesn't support named arguments.

Syntax

object.**Add(***index, key, text, width, alignment***)**

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

Part	Description
object	Required. An object expression that evaluates to a **ColumnHeaders** collection.
index	Optional. An integer that uniquely identifies a member of an object collection.
key	Optional. A unique string expression that can be used to access a member of the collection.
text	Optional. A string that appears in the **ColumnHeader** object.
width	Optional. A numeric expression specifying the width of the object using the scale units of the control's container.
alignment	Optional. An integer that determines the alignment of text in the **ColumnHeader** object. For settings, choose the **Alignment** property from the See Also list.

Remarks

The **Add** method returns a reference to the newly inserted **ColumnHeader** object.

Use the *index* argument to insert a column header in a specific position in the **ColumnHeaders** collection.

When the members of a **ColumnHeaders** collection can change dynamically, you may want to reference them using the **Key** property, because the **Index** property for any **ColumnHeader** object may be changing.

ListItem Object, ListItems Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjListItemC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjListItemX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vamthItem;vbobjListItemP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vamthClear;vamthRemove;vbobjListItemM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjListItemE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjListItemS"}

- A **ListItem** consists of text, the index of an associated icon (**ListImage** object), and, in Report view, an array of strings representing subitems.
- A **ListItems** collection contains one or more **ListItem** objects.

Syntax

listview.**ListItems**
listview.**ListItems(***index***)**

The syntax lines above refer to the collection and to individual elements in the collection, respectively, according to the standard collection syntax.

The **ListItem** object, **ListItems** collection syntax has these parts:

Part	Description
listview	An object expression that evaluates to a **ListView** control.
index	Either an integer or string that uniquely identifies a member of a **ListItem** collection. The integer is the value of the **Index** property; the string is the value of the **Key** property.

Remarks

ListItem objects can contain both text and pictures. However, to use pictures, you must reference an **ImageList** control using the **Icons** and **SmallIcons** properties.

You can also change the image by using the **Icon** or **SmallIcon** property.

The following example shows how to add **ColumnHeaders** and several **ListItem** objects with subitems to a **ListView** control.

```
Private Sub Form_Load()
   Dim clmX As ColumnHeader
   Dim itmX As ListItem
   Dim i As Integer

   For i = 1 To 3
      Set clmX = ListView1.ColumnHeaders.Add()
      clmX.Text = "Col" & i
   Next i

   ' Add 10 items to list, all with the same icon

   For i = 1 To 10
      Set itmX = ListView1.ListItems.Add()
      itmX.SmallIcon = 1
      itmX.Text = "ListItem " & i
      itmX.SubItems(1) = "Subitem 1"
      itmX.SubItems(2) = "Subitem 2"
   Next i
End Sub
```

Add Method (ListItems Collection)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthAddListItemsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAddListItemsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAddListItemsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAddListItemsS"}

Adds a **ListItem** object to a **ListItems** collection in a **ListView** control and returns a reference to the newly created object. Doesn't support named arguments.

Syntax

object.**Add(***index, key, text, icon, smallIcon***)**

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

Part	Description
object	Required. An object expression that evaluates to a **ListItems** collection.
index	Optional. An integer specifying the position where you want to insert the **ListItem**. If no index is specified, the **ListItem** is added to the end of the **ListItems** collection.
key	Optional. A unique string expression that can be used to access a member of the collection.
text	Optional. A string that is associated with the **ListItem** object control.
icon	Optional. An integer that sets the icon to be displayed from an **ImageList** control, when the **ListView** control is set to Icon view.
smallIcon	Optional. An integer that sets the icon to be displayed from an **ImageList** control, when the **ListView** control is set to SmallIcon view.

Remarks

Before setting either the **Icons** or **SmallIcons** properties, you must first initialize them. You can do this at design time by specifying an ImageList object with the General tab of the **ListView** Control Properties dialog box, or at run time with the following code:

```
ListView1.Icons = ImageList1   'Assuming the Imagelist is ImageList1.
ListView1.SmallIcons = ImageList2
```

If the list is not currently sorted, a **ListItem** object can be inserted in any position by using the *index* argument. If the list is sorted, the *index* argument is ignored and the **ListItem** object is inserted in the appropriate position based upon the sort order.

If *index* is not supplied, the **ListItem** object is added with an index that is equal to the number of **ListItem** objects in the collection + 1.

Use the **Key** property to reference a member of the **ListItems** collection if you expect the value of an object's **Index** property to change, such as by dynamically adding objects to or removing objects from the collection.

Add Method (ListItems, ColumnHeaders), ListItems Property, SubItems Property Example

The following example adds several **ListItem** objects with images from an **ImageList** control to a **ListView** control. To try this example, place a **ListView** control and two **ImageList** controls on a form and paste the code into the Declarations section.

Note The example will not run unless you add a reference to the Microsoft DAO 3.0 Object Library by using the References command on the Tools menu. Run the example.

```
Private Sub Form_Load()
   ' Create an object variable for the ColumnHeader object.
   Dim clmX As ColumnHeader
   ' Add ColumnHeaders. The width of the columns is the width
   ' of the control divided by the number of ColumnHeader objects.
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Author", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Author ID", ListView1.Width / 3, lvwColumnCenter)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Birthdate", ListView1.Width / 3)

   ListView1.View = lvwReport ' Set View property to Report.

   ' Load one image into an ImageList control.
   Dim imgX As ListImage
   Set imgX = ImageList1.ListImages.Add _
   (,,LoadPicture("icons\Writing\Note06.ico"))
   Set imgX = ImageList2.ListImages.Add _
   (,,LoadPicture("bitmaps\assorted\w.bmp"))
   ' Set Icons property to ImageList1.
   ListView1.Icons = ImageList1
   ListView1.SmallIcons = ImageList2

   ' Create object variables for the Data Access objects.
   Dim myDb As Database, myRs As Recordset
   ' Set the Database to the BIBLIO.MDB database.
   Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")
   ' Set the recordset to the "Authors" table.
   Set myRs = myDb.OpenRecordset("Authors", dbOpenDynaset)

   ' Create a variable to add ListItem objects.
   Dim itmX As ListItem

   ' While the record is not the last record, add a ListItem object.
   ' Use the author field for the ListItem object's text.
   ' Use the AuthorID field for the ListItem object's SubItem(1).
   ' Use the "Year of Birth" field for the ListItem object's SubItem(2).

   While Not myRs.EOF
      Set itmX = ListView1.ListItems. _
      Add(, , CStr(myRs!Author),1) ' Author.

      ' If the AuthorID field is not null, then set SubItem 1 to it.
      If Not IsNull(myRs!Au_id) Then
         itmX.SubItems(1) = CStr(myRs!Au_id)    ' Author ID.
      End If
```

```
      ' If the birth field is not Null, set the SubItem 2 to it.
      If Not IsNull(myRs![Year Born]) Then
         itmX.SubItems(2) = myRs![Year Born]
      End If
      myRs.MoveNext  ' Move to next record.
   Wend
End Sub
```

Alignment Property (ColumnHeader Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAlignmentCHC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAlignmentCHX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAlignmentCHA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAlignmentCHS"}

Returns or sets the alignment of text in a **ColumnHeader** object.

Syntax

object.**Alignment** [= *integer*]

The **Alignment** Property syntax has these parts:

Part	Description
object	An object expression that evaluates to a **ColumnHeader** object.
integer	An integer that determines the alignment, as described in Settings.

Settings

The settings for *integer* are:

Constant	Value	Description
lvwColumnLeft	0	(Default) Left. Text is aligned left.
lvwColumnRight	1	Right. Text is aligned right.
lvwColumnCenter	2	Center. Text is centered.

Arrange Property (ListView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproArrangeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproArrangeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproArrangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproArrangeS"}

Returns or sets a value that determines how the icons in a **ListView** control's Icon or SmallIcon view are arranged.

Syntax

object.**Arrange** [= *value*]

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

Part	Description
object	An object expression that evaluates to a **ListView** control.
value	An integer or constant that determines how the icons or small icons are arranged, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
lvwNoArrange	0	(Default) None.
lvwAutoLeft	1	Left. Items are aligned automatically along the left side of the control.
lvwAutoTop	2	Top. Items are aligned automatically along the top of the control.

Arrange Property Example

This example adds several **ListItem** objects and subitems to a **ListView** control. When you click on an **OptionButton** control, the **Arrange** property is set with the **Index** value of the **OptionButton**. To try the example, place a control array of three **OptionButton** controls, a **ListView** control, and two **ImageList** controls on a form and paste the code into the form's Declarations section. Run the example and click on an **OptionButton** to change the **Arrange** property.

```
Private Sub Option1_Click(Index as Integer)
   ' Set Arrange property to Option1.Index.
   ListView1.Arrange = Index
End Sub

Private Sub Form_Load()
   ' Label OptionButton controls with Arrange choices.
      Option1(0).Caption = "No Arrange"
      Option1(1).Caption = "Align Auto Left"
      Option1(2).Caption = "Align Auto Top"

   ' Declare variables for creating ListView and ImageList objects.
   Dim i As Integer
   Dim itmX As ListItem ' Object variable for ListItems.
   Dim imgX As ListImage   ' Object variable for ListImages.

   ' Add a ListImage object to an ImageList control.
   Set imgX = ImageList1.ListImages. _
   Add(,,LoadPicture("icons\mail\mail01a.ico"))

   ListView1.Icons = ImageList1    ' Associate an ImageList control.

   ' Add ten ListItem objects, each with an Icon.
   For i = 1 To 10
      Set itmX = ListView1.ListItems.Add()
      itmX.Icon = 1  ' Icon.
      itmX.Text = "ListItem " & i
   Next i
End Sub
```

ColumnClick Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtColumnClickC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtColumnClickX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtColumnClickA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtColumnClickS"}

Occurs when a **ColumnHeader** object in a **ListView** control is clicked. Only available in Report view.

Syntax

Private Sub *object*_**ColumnClick**(**ByVal** *columnheader* **As ColumnHeader**)

The **ColumnClick** event syntax has these parts:

Part	Description
object	An object expression that evaluates to a **ListView** control.
columnheader	A reference to the **ColumnHeader** object that was clicked.

Remarks

The **Sorted**, **SortKey**, and **SortOrder** properties are commonly used in code to sort the **ListItem** objects in the clicked column.

ColumnHeaders Property (ListView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproColumnHeadersC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproColumnHeadersX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproColumnHeadersA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproColumnHeadersS"}

Returns a reference to a collection of **ColumnHeader** objects.

Syntax

object.**ColumnHeaders**

The *object* placeholder represents an object expression that evaluates to a **ListView** control.

Remarks

You can manipulate **ColumnHeader** objects using standard collection methods (for example, the **Remove** method). Each **ColumnHeader** in the collection can be accessed either by its index or by a unique key, stored in the **Key** property.

Ghosted Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproGhostedC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproGhostedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproGhostedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproGhostedS"}

Returns or sets a value that determines whether a **ListItem** object in a **ListView** control is unavailable (it appears dimmed).

Syntax

*object.***Ghosted** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to a **ListItem** object.
boolean	A Boolean expression specifying if the icon or small icon is ghosted, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The **ListItem** object is unavailable to the user.
False	(Default) The **ListItem** is available.

Remarks

The **Ghosted** property is typically used to show when a **ListItem** is cut, or disabled for some reason.

When a ghosted **ListItem** is selected, the label is highlighted but its image is not.

Ghosted, MultiSelect Properties Example

This example populates a **ListView** control with the contents of the Authors table from the BIBLIO.MDB database, and lets you use **OptionButton** controls to set **MultiSelect** property options. You can select any item, or hold down the SHIFT Key and select multiple items. Clicking on the **CommandButton** sets the **Ghosted** property of the selected items to **True**. To try the example, place a control array of two **OptionButton** controls, a **ListView** control, an **ImageList** control, and a **CommandButton** control on a form and paste the code into the form's Declarations section.

Note The example will not run unless you add a reference to the Microsoft DAO 3.0 Object Library by using the References command on the Tools menu. Run the example, select a **MultiSelect** option by clicking an **OptionButton**, click on items to select them and click the **CommandButton** to ghost them.

```
Private Sub Command1_Click()
   Dim x As Object
   Dim i As Integer
   ' Ghost selected ListItem.
If ListView1.SelectedItem Is Nothing Then Exit Sub
   For i = 1 To ListView1.ListItems.Count
      If ListView1.ListItems(i).Selected = True Then
         ListView1.ListItems(i).Ghosted = True
      End If
   Next i
End Sub

Private Sub Form_Load()
   ' Create an object variable for the ColumnHeader object.
   Dim clmX As ColumnHeader
   ' Add ColumnHeaders. The width of the columns is the width
   ' of the control divided by the number of ColumnHeader objects.
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Company", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Address", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Phone", ListView1.Width / 3)

   ' Label OptionButton controls with MultiSelect options.
      Option1(0).Caption = "No MultiSelect"
      Option1(1).Caption = "MultiSelect"
      ListView1.MultiSelect = 1 ' Set MultiSelect to True

   ListView1.BorderStyle = ccFixedSingle ' Set BorderStyle property.
   ListView1.View = lvwReport      ' Set View property to Report.
   ' Add one image to ImageList control.
   Dim imgX As ListImage
   Set imgX = ImageList1.ListImages. _
   Add(, , LoadPicture("icons\mail\mail01a.ico"))
   ListView1.Icons = ImageList1

   ' Create object variables for the Data Access objects.
   Dim myDb As Database, myRs As Recordset
   ' Set the Database to the BIBLIO.MDB database.
   Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")
   ' Set the recordset to the Publishers table.
   Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)
```

```
   ' Create a variable to add ListItem objects.
   Dim itmX As ListItem

   ' While the record is not the last record, add a ListItem object.
   ' Use the Name field for the ListItem object's text.
   ' Use the Address field for the ListItem object's SubItem(1).
   ' Use the Phone field for the ListItem object's SubItem(2).

   While Not myRs.EOF
      Set itmX = ListView1.ListItems.Add(, , CStr(myRs!Name))
      itmX.Icon = 1  ' Set icon to the ImageList icon.

      ' If the Address field is not Null, set SubItem 1 to the field.
      If Not IsNull(myRs!Address) Then
         itmX.SubItems(1) = CStr(myRs!Address)  ' Address field.
      End If

      ' If the Phone field is not Null, set SubItem 2 to the field.
      If Not IsNull(myRs!Telephone) Then
         itmX.SubItems(2) = myRs!Telephone  ' Phone field.
      End If

      myRs.MoveNext  ' Move to next record.
   Wend

   ListView1.View = lvwIcon  ' Show Icons view.
   Command1.Caption = "Cut"  ' Set caption of the CommandButton.
   ' Add a caption to the form.
   Me.Caption = "Select any item(s) and click 'Cut'."
End Sub

Private Sub Option1_Click(Index as Integer)
   ListView1.MultiSelect = Index
End Sub
```

HideColumnHeaders Property (ListView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHideColumnHeadersC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHideColumnHeadersX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHideColumnHeadersA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHideColumnHeadersS"}

Returns or sets whether **ColumnHeader** objects in a **ListView** control are hidden in Report view.

Syntax

object.**HideColumnHeaders** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to a **ListView** control.
Boolean	A Boolean expression that specifies if the column headers are visible in Report view, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The column headers are not visible.
False	(Default) The column headers are visible.

Remarks

The **ListItem** objects and any related subitems remain visible even if the **HideColumnHeaders** property is set to **True**.

HideColumnHeaders Property Example

This example adds several **ListItem** objects with subitems to a **ListView** control. When you click on the **CommandButton**, the **HideColumnHeaders** property toggles between **True** (-1) and **False** (0). To try the example, place **ListView** and **CommandButton** controls on a form and paste the code into the form's Declarations section. Run the example and click the **CommandButton** to toggle the **HideColumnHeaders** property.

```
Private Sub Command1_Click()
   ' Toggle HideColumnHeaders property off and on.
   ListView1.HideColumnHeaders = Abs(ListView1.HideColumnHeaders) - 1
End Sub

Private Sub Form_Load()
   Dim clmX As ColumnHeader
   Dim itmX As ListItem
   Dim i As Integer
   Command1.Caption = "HideColumnHeaders"

   ' Add 3 ColumnHeader objects to the control.
   For i = 1 To 3
      Set clmX = ListView1.ColumnHeaders.Add()
      clmX.Text = "Col" & i
   Next I

   ' Set View to Report.
   ListView1.View = lvwReport

   ' Add 10 ListItems to the control.
   For i = 1 To 10
      Set itmX = ListView1.ListItems.Add()
      itmX.Text = "ListItem " & i
      itmX.SubItems(1) = "Subitem 1"
      itmX.SubItems(2) = "Subitem 2"
   Next i
End Sub
```

Icon, SmallIcon Properties (ListItem Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSmallIconC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSmallIconsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSmallIconA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSmallIcons"}

Returns or sets the index or key value of an icon or small icon associated with a **ListItem** object in an **ImageList** control.

Syntax

object.**Icon** [= *index*]
object.**SmallIcon** [= *index*]

The **Icon**, **SmallIcon** properties syntax has the following parts:

Part	Description
object	An object expression that evaluates to a **ListItem** object.
index	An integer or unique string that identifies an icon or small icon in an associated **ImageList** control. The integer is the value of the **ListItem** object's **Index** property; the string is the value of the **Key** property.

Remarks

Before you can use an icon in a **ListItem** object, you must associate an **ImageList** control with the **ListView** control containing the object. See the **Icons**, **SmallIcons** Properties (**ListView** Control) for more information. The example below shows the proper syntax:

```
ListView1.ListItems(1).SmallIcons=1
```

The images will appear when the **ListView** control is in SmallIcons view.

Icons, SmallIcons Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSmallIconsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSmallIconsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSmallIconsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSmallIconsS"}

Returns or sets the **ImageList** controls associated with the Icon and SmallIcon views in a **ListView** control.

Syntax

object.**Icons** [= *imagelist*]
object.**SmallIcons** [= *imagelist*]

The **Icons**, **SmallIcons** properties syntax has the following parts:

Part	Description
object	An object expression that evaluates to the **ListView** control.
imagelist	An object expression that evaluates to an **ImageList** control.

Remarks

To associate an **ImageList** control with a **ListView** control at run time, set these properties to the desired **ImageList** control.

Each **ListItem** object in the **ListView** control also has **Icon** and **SmallIcon** properties, which index the **ListImage** objects and determine which image is displayed.

Once you associate an **ImageList** with the **ListView** control, you can use the value of either the **Index** or **Key** property to refer to a **ListImage** object in a procedure.

Icon, SmallIcon, Icons, SmallIcons, View Properties Example

This example populates a **ListView** control with the contents of the Publishers table in the BIBLIO.MDB database. Four **OptionButton** controls are labeled with **View** property choices. You must place two **ImageList** controls on the form, one to contain images for the **Icon** property, and a second to contain images for the **SmallIcon** property of each **ListItem** object. To try the example, place a **ListView**, a control array of four **OptionButton** controls, and two **ImageList** controls on a form and paste the code into the form's Declarations section.

Note The example will not run unless you add a reference to the Microsoft DAO 3.0 Object Library by using the References command on the Tools menu. Run the example and click on the **ComboBox** control to switch views.

```
Private Sub Option1_Click(Index as Integer)
   ' Set the ListView control's View property to the
   ' Index of Option1
   ListView1.View = Index
End Sub

Private Sub Form_Load()
   ' Create an object variable for the ColumnHeader object.
   Dim clmX As ColumnHeader
   ' Add ColumnHeaders. The width of the columns is the width
   ' of the control divided by the number of ColumnHeader objects.
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Company", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Address", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Phone", ListView1.Width / 3)

   ListView1.BorderStyle = ccFixedSingle ' Set BorderStyle property.
   ListView1.View = lvwReport ' Set View property to Report.

   ' Add one image to ImageList1--the Icons ImageList.
   Dim imgX As ListImage
   Set imgX = ImageList1.ListImages. _
   Add(, , LoadPicture("icons\mail\mail01a.ico"))
   ' Add an image to ImageList2--the SmallIcons ImageList.
   Set imgX = ImageList2.ListImages. _
   Add(, , LoadPicture("bitmaps\assorted\w.bmp"))

   ' To use ImageList controls with the ListView control, you must
   ' associate a particular ImageList control with the Icons and
   ' SmallIcons properties.
   ListView1.Icons = ImageList1
   ListView1.SmallIcons = ImageList2
   ' Label OptionButton controls with View options.
      Option1(0).Caption = "Icon"
      Option1(1).Caption = "SmallIcon"
      Option1(2).Caption = "List"
      Option1(3).Caption = "Report"
      ListView1.View = lvwIcon ' Set to Icon view

   ' Create object variables for the Data Access objects.
   Dim myDb As Database, myRs As Recordset
   ' Set the Database to the BIBLIO.MDB database.
```

```
    Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")
    ' Set the recordset to the Publishers table.
    Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

    ' Create a variable to add ListItem objects.
    Dim itmX As ListItem

    ' While the record is not the last record, add a ListItem object.
    ' Use the Name field for the ListItem object's text.
    ' Use the Address field for the ListItem object's SubItem(1)
    ' Use the Phone field for the ListItem object's SubItem(2)

    While Not myRs.EOF

       Set itmX = ListView1.ListItems.Add(, , CStr(myRs!Name))
       itmX.Icon = 1  ' Set an icon from ImageList1.
       itmX.SmallIcon = 1   ' Set an icon from ImageList2.

       ' If the Address field is not Null, set SubItem 1 to the field.
       If Not IsNull(myRs!Address) Then
          itmX.SubItems(1) = CStr(myRs!Address) ' Address field.
       End If

       ' If the Phone field is not Null, set SubItem 2 to the field.
       If Not IsNull(myRs!Telephone) Then
          itmX.SubItems(2) = myRs!Telephone  ' Phone field.
       End If

       myRs.MoveNext  ' Move to next record.
    Wend
End Sub
```

ListItems Property (ListView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproListItemsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproListItemsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproListItemsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproListItemsS"}

Returns a reference to a collection of **ListItem** objects in a **ListView** control.

Syntax

object.**ListItems**

The *object* placeholder represents an object expression that evaluates to a **ListView** control.

Remarks

ListItem obects can be manipulated using the standard collection methods. Each **ListItem** in the collection can be accessed by its unique key, which you create and store in the **Key** property.

You can also retrieve **ListItem** objects by their display position using the **Index** property.

LabelWrap Property (ListView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLabelWrapC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLabelWrapX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLabelWrapA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLabelWrapS"}

Returns or sets a value that determines whether or not labels are wrapped when a **ListView** control is in Icon view.

Syntax

object.**LabelWrap** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to a **ListView** control.
boolean	A Boolean expression specifying if the labels wrap, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) The labels wrap.
False	The labels don't wrap.

Remarks

The length of the label is determined by setting the icon spacing in the Control Panel, Desktop option, in Windows NT. In Windows 95, use the Appearance tab in the Display control panel.

MultiSelect Property (ListView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMultiSelectListViewC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMultiSelectListViewX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMultiSelectListViewA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMultiSelectListViewS"}

Returns or sets a value indicating whether a user can select multiple **ListItems** in the **ListView** control.

Syntax

object.**MultiSelect** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to a **ListView** control.
boolean	A value specifying the type of selection, as described in Settings.

Settings

The settings for *boolean* are:

Constant	Description
False	(Default) Selecting multiple **ListItems** isn't allowed.
True	Multiple selection. Pressing SHIFT and clicking the mouse or pressing SHIFT and one of the arrow keys (UP ARROW, DOWN ARROW, LEFT ARROW, and RIGHT ARROW) extends the selection from the previously selected **ListItem** to the current **ListItem**. Pressing CTRL and clicking the mouse selects or deselects a **ListItem** in the list.

SortKey Property (ListView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSortKeyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSortKeyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSortKeyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSortKeyS"}

Returns or sets a value that determines how the **ListItem** objects in a **ListView** control are sorted.

Syntax

object.**SortKey** [= *integer*]

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

Part	Description
object	An object expression that evaluates to a **ListView** control.
integer	An integer specifying the sort key, as described in Settings.

Settings

The settings for *integer* are:

Setting	Description
0	Sort using the **ListItem** object's **Text** property.
≥ 1	Sort using the subitem whose collection index is specified here.

Remarks:

The **Sorted** property must be set to **True** before the change takes place.

It is common to sort a list when the column header is clicked. For this reason, the **SortKey** property is commonly included in the **ColumnClick** event to sort the list using the clicked column, as determined by the sort key, and demonstrated in the following example:

```
Private Sub ListView1_ColumnClick (ByVal ColumnHeader as ColumnHeader)
   ListView1.SortKey=ColumnHeader.Index-1
End Sub
```

SortOrder Property (ListView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSortOrderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSortOrderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSortOrderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSortOrderS"}

Returns or sets a value that determines whether **ListItem** objects in a **ListView** control are sorted in ascending or descending order.

Syntax

object.**SortOrder** [= *integer*]

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

Part	Description
object	An object expression that evaluates to a **ListView** control.
integer	An integer specifying the type of sort order, as described in Settings.

Settings

The settings for *integer* are:

Constant	Value	Description
lvwAscending	0	(Default) Ascending order. Sorts from the beginning of the alphabet (A-Z) or the earliest date. Numbers are sorted as strings, with the first digit determining the initial position in the sort, and subsequent digits determining sub-sorting.
lvwDescending	1	Descending order. Sorts from the end of the alphabet (Z-A) or the latest date. Numbers are sorted as strings, with the first digit determining the initial position in the sort, and subsequent digits determining sub-sorting.

Remarks

The **Sorted** property must be set to **True** before a list will be sorted in the order specified by **SortOrder**.

SortKey, SortOrder, Sorted Properties, ColumnClick Event Example

This example adds three **ColumnHeader** objects to a **ListView** control and populates the control with the Publishers records of the BIBLIO.MDB database. An array of two **OptionButton** controls offers the two choices for sorting records. When you click on a **ColumnHeader**, the **ListView** control is sorted according to the **SortOrder** property, as determined by the **OptionButtons**. To try the example, place a **ListView** and a control array of two **OptionButton** controls on a form and paste the code into the form's Declarations section. Run the example and click on the **ColumnHeaders** to sort, and click on the **OptionButton** to switch the **SortOrder** property.

Note the example will not run unless you add a reference to the Microsoft DAO 3.0 Object Library by using the References command on the Tools menu.

```
Private Sub Option1_Click(Index as Integer)
   ' These OptionButtons offer two choices: Ascending (Index 0),
   ' and Descending (Index 1). Clicking on one of these
   ' sets the SortOrder for the ListView control.
   ListView1.SortOrder = Index
   ListView1.Sorted = True ' Sort the List.
End Sub

Private Sub Form_Load()
   ' Create an object variable for the ColumnHeader object.
   Dim clmX As ColumnHeader
   ' Add ColumnHeaders. The width of the columns is the width
   ' of the control divided by the number of ColumnHeader objects.
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Company", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Address", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Phone", ListView1.Width / 3)

   ListView1.BorderStyle = ccFixedSingle ' Set BorderStyle property.
   ListView1.View = lvwReport ' Set View property to Report.

   ' Label OptionButton controls with SortOrder options.
      Option1(0).Caption = "Ascending (A-Z)"
      Option1(1).Caption = "Descending (Z-A)"
      ListView1.SortOrder = 0 ' Set to Icon view

   ' Create object variables for the Data Access objects.
   Dim myDb As Database, myRs As Recordset
   ' Set the Database to the BIBLIO.MDB database.
   Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")
   ' Set the recordset to the Publishers table.
   Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

   ' Create a variable to add ListItem objects.
   Dim itmX As ListItem

   ' While the record is not the last record, add a ListItem object.
   ' Use the Name field for the ListItem object's text.
   ' Use the Address field for the ListItem object's subitem(1).
   ' Use the Phone field for the ListItem object's subitem(2).

   While Not myRs.EOF
```

```
      Set itmX = ListView1.ListItems.Add(, , CStr(myRs!Name))

      ' If the Address field is not Null, set subitem 1 to the field.
      If Not IsNull(myRs!Address) Then
         itmX.SubItems(1) = CStr(myRs!Address)  ' Address field.
      End If

      ' If the Phone field is not Null, set subitem 2 to the field.
      If Not IsNull(myRs!Telephone) Then
         itmX.SubItems(2) = myRs!Telephone  ' Phone field.
      End If

      myRs.MoveNext  ' Move to next record.
   Wend
End Sub

Private Sub ListView1_ColumnClick(ByVal ColumnHeader As ColumnHeader)
   ' When a ColumnHeader object is clicked, the ListView control is
   ' sorted by the subitems of that column.
   ' Set the SortKey to the Index of the ColumnHeader - 1
   ListView1.SortKey = ColumnHeader.Index - 1
   ' Set Sorted to True to sort the list.
   ListView1.Sorted = True
End Sub
```

SubItemIndex Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSubItemIndexC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSubItemIndexX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSubItemIndexA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSubItemIndexS"}

Returns the index of the subitem associated with a **ColumnHeader** object in a **ListView** control.

Syntax

object.**SubItemIndex** [= *integer*]

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

Part	Description
object	An object expression that evaluates to a **ColumnHeader** object.
integer	An integer specifying the index of the subitem associated with the **ColumnHeader** object.

Remarks

Subitems are arrays of strings representing the **ListItem** object's data when displayed in Report view.

The first column header always has a **SubItemIndex** property set to 0 because the small icon and the **ListItem** object's text always appear in the first column and are considered **ListItem** objects rather than subitems.

The number of column headers dictates the number of subitems. There is always exactly one more column header than there are subitems.

SubItemIndex Property Example

This example adds three **ColumnHeader** objects to a **ListView** control. The code then adds several **ListItem** and **Subitems** using the **SubItemIndex** to associate the **SubItems** string with the correct **ColumnHeader** object. To try the example, place a **ListView** control on a form and paste the code into the form's Declarations section. Run the example.

```
' Make sure ListView control is in report view.
ListView1.View = lvwReport

' Add three columns.
ListView1.ColumnHeaders.Add , "Name", "Name"
ListView1.ColumnHeaders.Add , "Address", "Address"
ListView1.ColumnHeaders.Add , "Phone", "Phone"

' Add ListItem objects to the control.
Dim itmX As ListItem
' Add names to column 1.
Set itmX= ListView1.ListItems.Add(1, "Mary", "Mary")
' Use the SubItemIndex to associate the SubItem with the correct
' ColumnHeader. Use the key ("Address") to specify the right
' ColumnHeader.
itmX.SubItems(ListView1.ColumnHeaders("Address").SubItemIndex) _
= "212 Grunge Street"
' Use the ColumnHeader key to associate the SubItems string
' with the correct ColumnHeader.
itmX.SubItems(ListView1.ColumnHeaders("Phone").SubItemIndex) _
= "555-1212"

Set itmX = ListView1.ListItems.Add(2, "Bill", "Bill")
itmX.SubItems(ListView1.ColumnHeaders("Address").SubItemIndex) _
= "101 Pacific Way"
itmX.SubItems(ListView1.ColumnHeaders("Phone").SubItemIndex) _
= "555-7879"

Set itmX= ListView1.ListItems.Add(3, "Susan", "Susan")
itmX.SubItems(ListView1.ColumnHeaders("Address").SubItemIndex) = _
"800 Chicago Street"
itmX.SubItems(ListView1.ColumnHeaders("Phone").SubItemIndex) = _
"555-4537"

Set itmX= ListView1.ListItems.Add(4, "Tom", "Tom")
itmX.SubItems(ListView1.ColumnHeaders("Address").SubItemIndex) _
= "200 Ocean City"
itmX.SubItems(ListView1.ColumnHeaders("Phone").SubItemIndex) = _
"555-0348"
```

View Property (ListView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproViewC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproViewX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproViewA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproViewS"}

Returns or sets the appearance of the **ListItem** objects in a **ListView** control.

Syntax

object.**View** [= *value*]

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

Part	Description
object	The object expression that evaluates to a **ListView** control.
value	An integer or constant specifying the control's appearance, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
lvwIcon	0	(Default) Icon. Each **ListItem** object is represented by a full-sized (standard) icon and a text label.
lvwSmallIcon	1	SmallIcon. Each **ListItem** object is represented by a small icon and a text label that appears to the right of the icon. The items appear horizontally.
lvwList	2	List. Each **ListItem** object is represented by a small icon and a text label that appears to the right of the icon. The **ListItem** objects are arranged vertically, each on its own line with information arranged in columns.
lvwReport	3	Report. Each **ListItem** object is displayed with its small icon and text labels. You can provide additional information about each **ListItem** object in a subitem. The icons, text labels, and information appear in columns with the leftmost column containing the small icon, followed by the text label. Additional columns display the text for each of the item's subitems.

Remarks

In Icon view only, use the **LabelWrap** property to specify if the **ListItem** object's labels are wrapped or not.

In Report view, you can hide the column headers by setting the **HideColumnHeaders** property to

True. You can also use the ColumnClick event and the **Sorted**, **SortOrder**, and **SortKey** properties to sort the **ListItem** objects or subitems when a user clicks a column header. The user can change the size of the column by grabbing the right border of a column header and dragging it to the desired size.

FindItem Method (ListView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthFindItemC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthFindItemX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthFindItemA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthFindItemS"}

Finds and returns a reference to a **ListItem** object in a **ListView** control. Doesn't support named arguments.

Syntax

object.**FindItem** (*string*, *value*, *index*, *match*)

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

Part	Description
object	Required. An object expression that evaluates to a **ListView** control.
string	Required. A string expression indicating the **ListItem** object to be found.
value	Optional. An integer or constant specifying whether the string will be matched to the **ListItem** object's **Text**, **Subitems**, or **Tag** property, as described in Settings.
index	Optional. An integer or string that uniquely identifies a member of an object collection and specifies the location from which to begin the search. The integer is the value of the **Index** property; the string is the value of the **Key** property. If no index is specified, the default is 1.
match	Optional. An integer or constant specifying that a match will occur if the item's **Text** property is the same as the string, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
lvwText	0	(Default) Matches the string with a **ListItem** object's **Text** property.
lvwSubitem	1	Matches the string with any string in a **ListItem** object's **SubItems** property.
lvwTag	2	Matches the string with any **ListItem** object's **Tag** property.

The settings for *match* are:

Constant	Value	Description
lvwWholeWord	0	(Default) An integer or constant specifying that a match will occur if the item's **Text** property begins with the whole word being searched. Ignored if the criteria is not text.
lvwPartial	1	An integer or constant specifying

	that a match will occur if the item's **Text** property begins with the string being searched. Ignored if the criteria is not text.

Remarks

If you specify Text as the search criteria, you can use **lvwPartial** so that a match occurs when the **ListItem** object's **Text** property begins with the string you are searching for. For example, to find the **ListItem** whose text is "Autoexec.bat", use:

```
'Create a ListItem variable.
Dim itmX As ListItem
'Set the variable to the found item.
Set itmX = ListView1.FindItem("Auto",,,lvwpartial)
```

FindItem Method Example

This example populates a **ListView** control with the contents of the Publishers table of the BIBLIO.MDB database. A **ComboBox** control is also populated with three options for the **FindItem** method. A **CommandButton** contains the code for the **FindItem** method; when you click on the button, you are prompted to enter the string to search for, and the **FindItem** method searches the **ListView** control for the string. If the string is found, the control is scrolled using the **EnsureVisible** method to show the found **ListItem** object. To try the example, place a **ListView**, **ComboBox**, and a **CommandButton** control on a form and paste the code into the form's Declarations section. Run the example and click on the command button.

Note the example will not run unless you add a reference to the Microsoft DAO 3.0 Object Library by using the References command from the Tools menu.

```
Private Sub Form_Load()
   ' Create an object variable for the ColumnHeader object.
   Dim clmX As ColumnHeader
   ' Add ColumnHeaders. The width of the columns is the width
   ' of the control divided by the number of ColumnHeader objects.
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Company", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Address", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Phone", ListView1.Width / 3)

   ListView1.BorderStyle = ccFixedSingle  ' Set BorderStyle property.
   ListView1.View = lvwReport  ' Set View property to Report.
   Command1.Caption = "&FindItem"

   ' Label OptionButton controls with FindItem options.
      Option1(0).Caption = "Text"
      Option1(1).Caption = "SubItem"
      Option1(2).Caption = "Tag"
      ListView1.FindItem = 0 ' Set the ListView FindItem property to Text.
   End With

   ' Populate the ListView control with database records.
   ' Create object variables for the Data Access objects.
   Dim myDb As Database, myRs As Recordset
   ' Set the Database to the BIBLIO.MDB database.
   Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")
   ' Set the recordset to the Publishers table.
   Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

   ' While the record is not the last record, add a ListItem object.
   ' Use the reference to the new object to set properties.
   ' Set the Text property to the Name field (myRS!Name).
   ' Set SubItem(1) to the Address field (myRS!Address).
   ' Set SubItem(7) to the Phone field (myRS!Telephone).

   While Not myRs.EOF
      Dim itmX As ListItem ' A ListItem variable.
      Dim intCount As Integer      ' A counter variable.
      ' Use the Add method to add a new ListItem and set an object
      ' variable to the new reference. Use the reference to set
      ' properties.
```

```
      Set itmX = ListView1.ListItems.Add(, , CStr(myRs!Name))
      intCount = intCount + 1 ' Increment counter for the Tag property.
      itmX.Tag = "ListItem " & intCount  ' Set Tag with counter.

      ' If the Address field is not Null, set SubItem 1 to Address.
      If Not IsNull(myRs!Address) Then
         itmX.SubItems(1) = CStr(myRs!Address) ' Address field.
      End If

      ' If the Phone field is not Null, set SubItem 2 to Phone.
      If Not IsNull(myRs!Telephone) Then
         itmX.SubItems(2) = myRs!Telephone  ' Phone field.
      End If

      myRs.MoveNext  ' Move to next record.
   Wend
End Sub

Private Sub Command1_Click()
   ' FindItem method.
   'Create an integer variable called intSelectedOption
   ' to store the index of the selected button
   ' Create a string variable called strFindMe. Use the InputBox
   ' to store the string to be found in the variable. Use the
   ' FindItem method to find the string. Option1 is used to
   ' switch the FindItem argument that determines where to look.

   Dim intSelectedOption as Integer
   Dim strFindMe As String
   If Option1(0).Value = True then
      strFindMe = InputBox("Find in " & Option1(0).Caption)
      intSelectedOption = lvwText
   If Option1(1).Value = True then
      strFindMe = InputBox("Find in " & Option1(1).Caption)
      intSelectedOption = lvwSubItem
   If Option1(2).Value = True then
      strFindMe = InputBox("Find in " & Option1(2).Caption)
      intSelectedOption = lvwTag

   ' FindItem method returns a reference to the found item, so
   ' you must create an object variable and set the found item
   ' to it.
   Dim itmFound As ListItem  ' FoundItem variable.

   Set itmFound = ListView1. _
   FindItem(strFindMe, intSelectedOption, , lvwPartial)

   ' If no ListItem is found, then inform user and exit. If a
   ' ListItem is found, scroll the control using the EnsureVisible
   ' method, and select the ListItem.
   If itmFound Is Nothing Then  ' If no match, inform user and exit.
      MsgBox "No match found"
      Exit Sub
   Else
       itmFound.EnsureVisible ' Scroll ListView to show found ListItem.
       itmFound.Selected = True    ' Select the ListItem.
      ' Return focus to the control to see selection.
```

```
        ListView1.SetFocus
    End If
End Sub

Private Sub ListView1_LostFocus()
    ' After the control loses focus, reset the Selected property
    ' of each ListItem to False.
    Dim i As Integer
    For i = 1 to ListView1.ListItems.Count
       ListView1.ListItems.Item(i).Selected = False
    Next i
End Sub
```

GetFirstVisible Method (ListView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetFirstVisibleC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetFirstVisibleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetFirstVisibleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetFirstVisibleS"}

Returns a reference to the first **ListItem** object visible in the internal area of a **ListView** control.

Syntax

object.**GetFirstVisible()**

The *object* placeholder represents an object expression that evaluates to a **ListView** control.

Remarks

A **ListView** control can contain more **ListItem** objects than can be seen in the internal area of the **ListView** control. You can use the reference returned by the **GetFirstVisible** method to determine the first visible **ListItem** object in List or Report view.

GetFirstVisible Method Example

This example populates a **ListView** control with the contents of the Publishers table in the BIBLIO.MDB database. When you click on the **CommandButton** control, the text of the first visible item is displayed. Click on the column headers to change the **SortKey** property and click the **CommandButton** again. To try the example, place a **ListView** and a **CommandButton** control on a form and paste the code into the form's Declarations section.

Note the example will not run unless you add a reference to the Microsoft DAO 3.0 Object Library using the References command from the Tools menu. Run the example.

```
Private Sub Command1_Click()
   ' Create a ListItem variable and set the variable to the object
   ' returned by the GetFirstVisible method. Use the reference to
   ' display the text of the ListItem.
   Dim itmX As ListItem
   Set itmX = ListView1.GetFirstVisible
   MsgBox itmX.Text
End Sub

Private Sub Form_Load()
   ' Create an object variable for the ColumnHeader object.
   Dim clmX As ColumnHeader
   ' Add ColumnHeaders. The width of the columns is the width
   ' of the control divided by the number of ColumnHeader objects.
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Company", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Address", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Phone", ListView1.Width / 3)

   ListView1.BorderStyle = ccFixedSingle ' Set BorderStyle property.

   ' Create object variables for the Data Access objects.
   Dim myDb As Database, myRs As Recordset
   ' Set the Database to the BIBLIO.MDB database.
   Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")
   ' Set the recordset to the Publishers table.
   Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

   ' Create a variable to add ListItem objects.
   Dim itmX As ListItem

   ' While the record is not the last record, add a ListItem object.
   ' Use the Name field for the ListItem object's text.
   ' Use the Address field for the ListItem object's subitem(1).
   ' Use the Phone field for the ListItem object's subitem(2).

   While Not myRs.EOF

      Set itmX = ListView1.ListItems.Add(, , CStr(myRs!Name))

      ' If the Address field is not Null, set SubItem 1 to the field.
      If Not IsNull(myRs!Address) Then
         itmX.SubItems(1) = CStr(myRs!Address) ' Address field.
      End If
```

```
      ' If the Phone field is not Null, set the SubItem 2 to the field.
      If Not IsNull(myRs!Telephone) Then
         itmX.SubItems(2) = myRs!Telephone  ' Phone field.
      End If

      myRs.MoveNext  ' Move to next record.
   Wend
   ListView1.View = lvwReport      ' Set view to Report.
End Sub

Private Sub ListView1_ColumnClick(ByVal ColumnHeader As ColumnHeader)
   ListView1.SortKey = ColumnHeader.Index - 1
   ListView1.Sorted = True
End Sub
```

SubItems Property (ListView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSubItemsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSubitemsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSubitemsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSubitemsS"}

Returns or sets an array of strings (a subitem) representing the **ListItem** object's data in a **ListView** control.

Syntax

object.**SubItems(***index***)** [= *string*]

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

Part	Description
object	An object expression that evaluates to a **ListItem** object.
index	An integer that identifies a subitem for the specified **ListItem**.
string	Text that describes the subitem.

Remarks

Subitems are arrays of strings representing the **ListItem** object's data that are displayed in Report view. For example, you could show the file size and the date last modified for a file.

A **ListItem** object can have any number of associated item data strings (subitems) but each **ListItem** object must have the same number of subitems.

There are corresponding column headers defined for each subitem.

You cannot add elements directly to the subitems array. Use the **Add** method of the **ColumnHeaders** collection to add subitems.

ListView Control Constants

Listview Control Constants

Constant	Value	Description
lvwIcon	0	(Default) Icon. Each **ListItem** object is represented by a full-sized (standard) icon and a text label.
lvwSmallIcon	1	SmallIcon. Each **ListItem** is represented by a small icon and a text label that appears to the right of the icon. The items appear horizontally.
LvwList	2	List. Each **ListItem** is represented by a small icon and a text label that appears to the right of the icon. Each **ListItem** appears vertically and on its own line with information arranged in columns.
lvwReport	3	Report. Each **ListItem** is displayed with its small icons and text labels. You can provide additional information about each **ListItem**. The icons, text labels, and information appear in columns with the leftmost column containing the small icon, followed by the text label. Additional columns display the text for each of the item's subitems.

ListArrange Constants

Constant	Value	Description
lvwNoArrange	0	(Default) None.
lvwAutoLeft	1	Left. **ListItem** objects are aligned along the left side of the control.
lvwTop	2	Top. **ListItem** objects are aligned along the top of the control.

ListColumnAlignment Constants

Constant	Value	Description
lvwColumnLeft	0	(Default) Left. Text is aligned left.
lvwColumnRight	1	Right. Text is aligned right.
lvwColumnCenter	2	Center. Text is centered.

ListLabelEdit Constants

Constant	Value	Description
lvwAutomatic	0	(Default) Automatic. The BeforeLabelEdit event is generated when the user clicks the label of a selected node.
lvwManual	1	Manual. The BeforeLabelEdit event will be generated only when the **StartLabelEdit** method is invoked.

ListSortOrder Constants

Constant	Value	Description
lvwAscending	0	(Default) Ascending order. Sorts from the beginning of the alphabet (A-Z), the earliest date, or the lowest number.
lvwDescending	1	Descending order. Sorts from the end of the alphabet (Z-A), the latest date, or the highest number.

ListFindItemWhere Constants

Constant	Value	Description
lvwText	0	(Default) Text. Matches the string with a **ListItem** object's **Text** property.
lvwSubItem	1	SubItem. Matches the string with any string in a **ListItem** object's **SubItems** property.
lvwTag	2	Tag. Matches the string with any **ListItem** object's **Tag** property.

ListFindItemHow Constants

Constant	Value	Description
lvwWholeWord	0	(Default) Whole word. Sets the search so that a match occurs if the item's **Text** property begins with the whole word being searched for. Ignored if the criteria is not text.
lvwPartial	1	Partial. Sets the search so that a match occurs if the item's **Text** property begins with the string being searched for. Ignored if the criteria is not text.

Sorted Property (ListView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSortedListViewC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSortedListViewX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSortedListViewA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSortedListViewS"}

Returns or sets a value that determines whether the **ListItem** objects in a **ListView** control are sorted.

Syntax

object.**Sorted** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to a **ListView** control.
boolean	A Boolean expression specifying whether the **ListItem** objects are sorted, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The list items are sorted alphabetically, according to the **SortOrder** property.
False	The list items are not sorted.

Remarks

The **Sorted** property must be set to **True** for the settings in the **SortOrder** and **SortKey** properties to take effect.

Each time the coordinates of a **ListItem** change, the **Sorted** property becomes **False**.

ItemClick Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtItemclickC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtItemclickX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtItemclickA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtItemclickS"}

Occurs when a **ListItem** object in a **ListView** control is clicked.

Syntax

Private Sub *object*_**ItemClick(ByVal** *Item* **As ListItem)**

The **ItemClick** event syntax has these parts:

Part	Description
object	An object expression that evaluates to a **ListView** control.
listitem	The **ListItem** object that was clicked.

Remarks

Use this event to determine which **ListItem** was clicked. This event is triggered before the **Click** event. The standard **Click** event is generated if the mouse is clicked on any part of the **ListView** control. The **ItemClick** event is generated only when the mouse is clicked on the text or image of a **ListItem** object.

ItemClick Event Example

This example populates a **ListView** control with contents of the Publishers table in the BIBLIO.MDB database. When a **ListItem** object is clicked, the code checks the value of the **Index** property. If the value is less than 15, nothing occurs. If the value is greater than 15, the **ListItem** object is ghosted. To try the example, place a **ListView** control on a form and paste the code into the form's Declarations section. Run the example and click on one of the items.

```
Private ListView1_ItemClick(ByVal Item As ListItem)
   Select Case Item.Index
   Case Is = <15
      Exit Sub
   Case Is => 15
      ' Toggle Ghosted property.
      Item.Ghosted = Abs(Item.Ghosted) - 1
   End Select
End Sub

Private Sub Form_Load()
   ' Create an object variable for the ColumnHeader object.
   Dim clmX As ColumnHeader
   ' Add ColumnHeaders. The width of the columns is the width
   ' of the control divided by the number of ColumnHeader objects.
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Company", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Address", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Phone", ListView1.Width / 3)

   ListView1.BorderStyle = ccFixedSingle ' Set BorderStyle property.

   ' Create object variables for the Data Access objects.
   Dim myDb As Database, myRs As Recordset
   ' Set the Database to the BIBLIO.MDB database.
   Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")
   ' Set the recordset to the Publishers table.
   Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

   ' Create a variable to add ListItem objects.
   Dim itmX As ListItem

   ' While the record is not the last record, add a ListItem object.
   ' Use the Name field for the ListItem object's text.
   ' Use the Address field for the ListItem object's SubItem(1).
   ' Use the Phone field for the ListItem object's SubItem(2).

   While Not myRs.EOF

      Set itmX = ListView1.ListItems.Add(, , CStr(myRs!Name))

      ' If the Address field is not Null, set SubItem 1 to the field.
      If Not IsNull(myRs!Address) Then
         itmX.SubItems(1) = CStr(myRs!Address) ' Address field.
      End If

      ' If the Phone field is not Null, set the SubItem 2 to the field.
```

```
      If Not IsNull(myRs!Telephone) Then
         itmX.SubItems(2) = myRs!Telephone  ' Phone field.
      End If

      myRs.MoveNext ' Move to next record.
   Wend
   ListView1.View = lvwReport      ' Set View to Report.
End Sub

Private Sub ListView1_ColumnClick(ByVal ColumnHeader As ColumnHeader)
   ListView1.SortKey = ColumnHeader.Index - 1
   ListView1.Sorted = True
End Sub
```

ProgressBar Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjProgressbarC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjProgressbarX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjProgressbarP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjProgressbarM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjProgressbarE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjProgressbarS"}

The **ProgressBar** control shows the progress of a lengthy operation by filling a rectangle with chunks from left to right.

Syntax

ProgressBar

Remarks

- The **ProgressBar** control monitors an operation's progress toward completion.

A **ProgressBar** control has a range and a current position. The range represents the entire duration of the operation. The current position represents the progress the application has made toward completing the operation. The **Max** and **Min** properties set the limits of the range. The **Value** property specifies the current position within that range. Because chunks are used to fill in the control, the amount filled in only approximates the **Value** property's current setting. Based on the control's size, the **Value** property determines when to display the next chunk.

The **ProgressBar** control's **Height** and **Width** properties determine the number and size of the chunks that fill the control. The more chunks, the more accurately the control portrays an operation's progress. To increase the number of chunks displayed, decrease the control's **Height** or increase its **Width**. The **BorderStyle** property setting also affects the number and size of the chunks. To accommodate a border, the chunk size becomes smaller.

You can use the **Align** property with the **ProgressBar** control to automatically position it at the top or bottom of the form.

Tip To shrink the chunk size until the progress increments most closely match actual progress values, make the **ProgressBar** control at least 12 times wider than its height.

The following example shows how to use the **ProgressBar** control, named ProgressBar1, to show the progress of a lengthy operation of a large array. Put a **CommandButton** control and a **ProgressBar** control on a form. The **Align** property in the sample code positions the **ProgressBar** control along the bottom of the form. The **ProgressBar** control displays no text.

```
Private Sub Command1_Click()
   Dim Counter As Integer
   Dim Workarea(250) As String
   ProgressBar1.Min = LBound(Workarea)
   ProgressBar1.Max = UBound(Workarea)
   ProgressBar1.Visible = True

'Set the Progress's Value to Min.
   ProgressBar1.Value = ProgressBar1.Min

'Loop through the array.
   For Counter = LBound(Workarea) To UBound(Workarea)
      'Set initial values for each item in the array.
      Workarea(Counter) = "Initial value" & Counter
      ProgressBar1.Value = Counter
   Next Counter
   ProgressBar1.Visible = False
   ProgressBar1.Value = ProgressBar1.Min
```

```
End Sub

Private Sub Form_Load()
   ProgressBar1.Align = vbAlignBottom
   ProgressBar1.Visible = False
   Command1.Caption = "Initialize array"
End Sub
```

Distribution Note The **ProgressBar** control is part of a group of ActiveX controls that are found in the COMCTL32.OCX file. To use the **ProgressBar** control in your application, you must add the COMCTL32.OCX file to the project. When distributing your application, install the COMCTL32.OCX file in the user's Microsoft Windows System or System32 directory. For more information on how to add an ActiveX control to a project, see the *Programmer's Guide*.

Slider Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjSliderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjSliderX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjSliderP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjSliderM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjSliderE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjSliderS"}

A **Slider** control is a window containing a slider and optional tick marks. You can move the slider by dragging it, clicking the mouse to either side of the slider, or using the keyboard.

Syntax

Slider

Remarks

Slider controls are useful when you want to select a discrete value or a set of consecutive values in a range. For example, you could use a **Slider** to set the size of a displayed image by moving the slider to a given tick mark rather than by typing a number. To select a range of values, set the **SelectRange** property to **True**, and program the control to select a range when the SHIFT key is down.

The **Slider** control can be oriented either horizontally or vertically.

Distribution Note To use the **Slider** control in your application, you must add the COMCTL32.OCX file to the project. When distributing your application, install the COMCTL32.OCX file in the user's Microsoft Windows System or System32 directory. For more information on how to add an ActiveX control to a project, see the *Programmer's Guide*.

LargeChange, SmallChange Properties (Slider Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLargeChangeSliderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLargeChangeSliderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLargeChangeSliderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLargeChangeSliderS"}

- The **LargeChange** property sets the number of ticks the slider will move when you press the PAGEUP or PAGEDOWN keys, or when you click the mouse to the left or right of the slider.
- The **SmallChange** property sets the number of ticks the slider will move when you press the left or right arrow keys.

Syntax

object.**LargeChange** = *number*
object.**SmallChange** = *number*

The **LargeChange** and **SmallChange** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to a **Slider** control.
number	A Long integer specifying how many ticks the slider moves.

Remarks

The default for the **LargeChange** property is 5. The default for the **SmallChange** property is 1.

LargeChange, SmallChange Properties Example

This example matches a **TextBox** control's width to that of a **Slider** control. While the **Slider** control's **Value** property is above a certain value, the **TextBox** control's width matches the **Slider** control's value. The **SmallChange** and **LargeChange** properties depend on the value of the **Slider** control's **Max** property. To try the example, place a **Slider** control and a **TextBox** control on a form and paste the code into the form's Declarations section. Run the example and press the PAGEDOWN, PAGEUP, and LEFT and RIGHT ARROW keys.

```
Private Sub Form_Load()
   Text1.Width = 4500 ' Set a minimum width for the TextBox.
   Slider1.Left = Text1.Left ' Align the Slider to the TextBox.
   ' Match the width of the Slider to the TextBox.
   Slider1.Max = Text1.Width
   ' Place the Slider a little below the Textbox.
   Slider1.Top = Text1.Top + Text1.Height + 50
   ' Set TickFrequency to a fraction of the Max value.
   Slider1.TickFrequency = Slider1.Max * 0.1
   ' Set LargeChange and SmallChange value to a fraction of Max.
   Slider1.LargeChange = Slider1.Max * 0.1
   Slider1.SmallChange = Slider1.Max * 0.01
End Sub

Private Sub Slider1_Change()
   ' If the slider is under 1/3 the size of the textbox, no change.
   ' Else, match the width of the textbox to the Slider's value.
   If Slider1.Value > Slider1.Max / 3 Then
      Text1.Width = Slider1.Value
   End If
End Sub
```

Orientation Property (Slider Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOrientationSliderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOrientationSliderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOrientationSliderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOrientationSliderS"}

Sets a value that determines whether the **Slider** control is oriented horizontally or vertically.

Syntax

object.**Orientation** = *number*

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

Part	Description
object	An object expression that evaluates to a **Slider** control.
number	A constant or value specifying the orientation, as described in Settings.

Settings

The settings for *number* are:

Constant	Value	Description
sldHorizontal	0	(Default) Horizontal. The slider moves horizontally and tick marks can be placed on either the top or bottom, both, or neither.
sldVertical	1	Vertical. The slider moves vertically and tick marks can be placed on either the left or right sides, both, or neither.

Orientation Property Example

This example toggles the orientation of a **Slider** control on a form. To try the example, place a **Slider** control onto a form and paste the code into the form's Declarations section, and then run the example. Click the form to toggle the **Slider** control's orientation.

```
Private Sub Form_Click()
   If Slider1.Orientation = 0 Then
      Slider1.Orientation = 1
   Else
      Slider1.Orientation = 0
   End If
End Sub
```

Scroll Event (Slider Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vafctMsgBox;vbevtScrollSliderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtScrollSliderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtScrollSliderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtScrollSliderS"}

Occurs when you move the slider on a **Slider** control, either by clicking on the control or using keyboard commands.

Syntax

Private Sub *object*_**Scroll( )**

The object placeholder represents an object expression that evaluates to a **Slider** control.

Remarks

The Scroll Event occurs before the Click event.

The Scroll Event continuously returns the value of the **Value** property as the slider is moved. You can use this event to perform calculations to manipulate controls that must be coordinated with ongoing changes in the **Slider** control. In contrast, use the Change event when you want an update to occur only once, after a **Slider** control's **Value** property has changed.

Note Avoid using a **MsgBox** statement or function in this event.

SelectRange Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSelectRangeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelectRangeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelectRangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelectRangeS"}

Sets a value that determines if a **Slider** control can have a selected range.

Syntax

object.**SelectRange** = *boolean*

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

Part	Description
object	An object expression that evaluates to a **Slider** control.
boolean	A Boolean expression that determines whether or not the **Slider** can have a selected range, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The **Slider** can have a selected range.
False	The **Slider** can't have a selected range.

Remarks

If **SelectRange** is set to **False**, then the **SelStart** property setting is the same as the **Value** property setting. Setting the **SelStart** property also changes the **Value** property, and vice-versa, which will be reflected in the position of the slider on the control. Setting **SelLength** when the **SelectRange** property is **False** has no effect.

SelectRange Property Example

This example allows the user to select a range when the SHIFT key is held down. To try the example, place a **Slider** control on a form and paste the code into the form's Declarations section. Run the example and select a range by holding down the SHIFT key and dragging or clicking the mouse on the **Slider** control.

```
Private Sub Form_Load()
   'Set slider control settings
   Slider1.Max = 20
End Sub

Private Sub Slider1_MouseDown(Button As Integer, Shift As Integer, x As
Single, y As Single)
   If Shift = 1 Then    ' If Shift button is down then
      Slider1.SelectRange = True   ' turn SelectRange on.
      Slider1.SelStart = Slider1.Value  ' Set the SelStart value
      Slider1.SelLength = 0 ' Set previous SelLength (if any) to 0.
   End If
End Sub

Private Sub Slider1_MouseUp(Button As Integer, Shift As Integer, x As
Single, y As Single)

   If Shift = 1 Then
   ' If user selects backwards from a point, an error will occur.
   On Error Resume Next
   ' Else set SelLength using SelStart and current value.
      Slider1.SelLength = Slider1.Value - Slider1.SelStart
   Else
      Slider1.SelectRange = False ' If user lifts SHIFT key.
   End If
End Sub
```

SelLength, SelStart Properties (Slider Control)

{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 length of a selected range in a **Slider** control.
- **SelStart** returns or sets the start of a selected range in a **Slider** control.

Syntax

object.**SelLength** [= *value*]
object.**SelStart** [= *value*]

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

Part	Description
object	An object expression that evaluates to a **Slider** control.
value	A value that falls within the **Min** and **Max** properties.

Remarks

The **SelLength** and **SelStart** properties are used together to select a range of contiguous values on a **Slider** control. The **Slider** control then has the additional advantage of being a visual analog of the range of possible values.

The **SelLength** property can't be less than 0, and the sum of **SelLength** and **SelStart** can't be greater than the **Max** property.

SelLength, SelStart Properties Example

This example selects a range on a **Slider** control. To try this example, place a **Slider** control onto a form with three **TextBox** controls, named Text1, Text2, and Text3. The **Slider** control's **SelectRange** property must be set to **True**. Paste the code below into the form's Declarations section, and run the example. While holding down the SHIFT key, you can select a range on the slider, and the various values will be displayed in the text boxes.

```
Private Sub Form_Load()
   ' Make sure SelectRange is True so selection can occur.
   Slider1.SelectRange = True
End Sub

Private Sub Slider1_MouseDown(Button As Integer, Shift As Integer, x As
Single, y As Single)
   If Shift = 1 Then ' If SHIFT is down, begin the range selection.
      Slider1.ClearSel  ' Clear any previous selection.
      Slider1.SelStart = Slider1.Value
      Text2.Text = Slider1.SelStart   ' Show the beginning
            ' of the range in the textbox.
   Else
      Slider1.ClearSel  ' Clear any previous selection.
   End If
End Sub

Private Sub Slider1_MouseUp(Button As Integer, Shift As Integer, x As
Single, y As Single)
   ' When SHIFT is down and SelectRange is True,
   ' this event is triggered.
   If Shift = 1 And Slider1.SelectRange = True Then
      ' Make sure the current value is larger than SelStart or
      ' an error will occur--SelLength can't be negative.
      If Slider1.Value >= Slider1.SelStart Then
         Slider1.SelLength = Slider1.Value - Slider1.SelStart
         Text1.Text = Slider1.Value  ' To see the end of the range.
         ' Text3 is the difference between the end and start values.
         Text3.Text = Slider1.SelLength
      End If
   End If
End Sub
```

TickFrequency Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTickFrequencyC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTickFrequencyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTickFrequencyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTickFrequencyS"}

Returns or sets the frequency of tick marks on a **Slider** control in relation to its range. For example, if the range is 100, and the **TickFrequency** property is set to 2, there will be one tick for every 2 increments in the range.

Syntax

object.**TickFrequency** [= *number*]

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

Part	Description
object	An object expression that evaluates to a **Slider** control.
number	A numeric expression specifying the frequency of tick marks.

TickFreqency Property Example

This example matches a **TextBox** control's width to that of a **Slider** control. While the **Slider** control's **Value** property is above a certain value, the **TextBox** control's width matches the **Slider** control's value. The **TickFrequency** depends on the value of the **Slider** control's **Max** property. To try the example, place a **Slider** and a **TextBox** control on a form and paste the code into the form's Declarations section. Run the example and click the slider several times.

```
Private Sub Form_Load()
   Text1.Width = 4500 ' Set a minimum width for the TextBox.
   Slider1.Left = Text1.Left ' Align the Slider to the TextBox.
   ' Match the width of the Slider to the TextBox.
   Slider1.Max = Text1.Width
   ' Place the Slider a little below the Textbox.
   Slider1.Top = Text1.Top + Text1.Height + 50
   ' Set TickFrequency to a fraction of the Max value.
   Slider1.TickFrequency = Slider1.Max * 0.1
   ' Set LargeChange and SmallChange value to a fraction of Max.
   Slider1.LargeChange = Slider1.Max * 0.1
   Slider1.SmallChange = Slider1.Max * 0.01
End Sub

Private Sub Slider1_Change()
   ' If the slider is under 1/3 the size of the textbox, no change.
   ' Else, match the width of the textbox to the Slider's value.
   If Slider1.Value > Slider1.Max / 3 Then
      Text1.Width = Slider1.Value
   End If
End Sub
```

TickStyle Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTickStyleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTickStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTickStyleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTickStyleS"}

Returns or sets the style (or positioning) of the tick marks displayed on the **Slider** control.

Syntax

object.**TickStyle** [= *number*]

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

Part	Description
object	An object expression that evaluates to a **Slider** control.
number	A constant or integer that specifies the **TickStyle** property, as described in Settings.

Settings

The settings for *number* are:

Constant	Value	Description
sldBottomRight	0	(Default) Bottom/Right. Tick marks are positioned along the bottom of the **Slider** if the control is oriented horizontally, or along the right side if it is oriented vertically.
sldTopLeft	1	Top/Left. Tick marks are positioned along the top of the **Slider** if the control is oriented horizontally, or along the left side if it is oriented vertically.
sldBoth	2	Both. Tick marks are positioned on both sides or top and bottom of the **Slider**.
sldNoTicks	3	None. No tick marks appear on the **Slider**.

TickStyle Property Example

This example allows you to see the various tick styles available in a drop-down list. To try the example, place a **Slider** control and a **ComboBox** control on a form. Paste the code into the Declarations section of the form, and run the example. Click on the **ComboBox** to change the **TickStyle** property value.

```
Private Sub Form_Load()
   With combo1
      .AddItem "Bottom/Right"
      .AddItem "Top/Left"
      .AddItem "Both"
      .AddItem "None"
      .ListIndex = 0
   End With
End Sub

Private Sub combo1_Click()
   Slider1.TickStyle = combo1.ListIndex
End Sub
```

ClearSel Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthClearSelC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthClearSelX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthClearSelA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthClearSelS"}

Clears the current selection of a **Slider** control.

Syntax

object.**ClearSel**

The object placeholder represents an object expression that evaluates to a **Slider** control.

Remarks

This method sets the **SelStart** property to the value of the **Value** property and sets the **SelLength** property to 0.

GetNumTicks Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetNumTicksC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetNumTicksX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetNumTicksA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetNumTicksS"}

Returns the number of ticks between the **Min** and **Max** properties of the **Slider** control.

Syntax

object.**GetNumTicks**

The object placeholder represents an object expression that evaluates to a **Slider** control.

Remarks

To change the number of ticks, reset the **Min** or **Max** properties or the **TickFrequency** property.

GetNumTicks Method Example

This example displays the current number of ticks on a **Slider** control, then increments the **Max** property by 10. To try this example, place a **Slider** control onto a form and paste the code into the form's Declarations section. Run the example, and click the **Slider** control to get the number of ticks. Every click on the control increases the ticks.

```
Private Sub Slider1_Click()
   MsgBox Slider1.GetNumTicks
   Slider1.Max = Slider1.Max + 10
End Sub
```

Slider Constants

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

Orientation Constants

Constant	Value	Description
sldHorizontal	0	Horizontal orientation.
sldVertical	1	Vertical orientation.

Tickstyle Constants

Constant	Value	Description
sldBottomRight	0	Bottom/Right. Tick marks are positioned along the bottom of the **Slider** if the control is oriented horizontally, or along the right side if it is oriented vertically.
sldTopLeft	1	Top/Left. Tick marks are positioned along the top of the **Slider** if the control is oriented horizontally, or along the left side if it is oriented vertically.
sldBoth	2	Both. Tick marks are positioned on both sides or top and bottom of the **Slider**.
sldNoTicks	3	None. No tick marks appear on the **Slider**.

StatusBar Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjStatusC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjStatusBarX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjStatusP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjStatusM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjStatusE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjStatusBarS"}

A **StatusBar** control provides a window, usually at the bottom of a parent form, through which an application can display various kinds of status data. The **StatusBar** can be divided up into a maximum of sixteen **Panel** objects that are contained in a **Panels** collection.

Syntax

StatusBar

Remarks

A **StatusBar** control consists of **Panel** objects, each of which can contain text and/or a picture. Properties to control the appearance of individual panels include **Width**, **Alignment** (of text and pictures), and **Bevel**. Additionally, you can use one of seven values of the **Style** property to automatically display common data such as date, time, and keyboard states.

At design time, you can create panels and customize their appearance by setting values in the Panel Properties dialog box. At run time, the **Panel** objects can be reconfigured to reflect different functions, depending on the state of the application. For detailed information about the properties, events, and methods of **Panel** objects, see the **Panel** Object and **Panels** Collection topics.

A **StatusBar** control typically displays information about an object being viewed on the form, the object's components, or contextual information that relates to that object's operation. The **StatusBar**, along with other controls such as the **Toolbar** control, gives you the tools to create an interface that is economical and yet rich in information.

Distribution Note The **StatusBar** control is part of a group of custom controls that are found in the COMCTL32.OCX file. To use the **StatusBar** control in your application, you must add the COMCTL32.OCX file to the project. When distributing your application, install the COMCTL32.OCX file in the user's Microsoft Windows SYSTEM directory. For more information on how to add a custom control to a project, see the *Programmer's Guide*.

StatusBar Control Constants

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

StatusBar Style Constants

Constant	Value	Description
sbrNormal	0	Normal. **StatusBar** is divided into panels.
sbrSimple	1	Simple. **StatusBar** has only one large panel and SimpleText.

Panel Alignment Constants

Constant	Value	Description
sbrLeft	0	Text to left.
sbrCenter	1	Text centered.
sbrRight	2	Text to right.

Panel Autosize Constants

Constant	Value	Description
sbrNoAutoSize	0	No Autosizing.
sbrSpring	1	Extra space divided among panels.
sbrContents	2	Fit to contents.

Panel Bevel Constants

Constant	Value	Description
sbrNoBevel	0	No bevel.
sbrInset	1	Bevel inset.
sbrRaised	2	Bevel raised.

Panel Style Constants

Constant	Value	Description
sbrText	0	Text and/or bitmap displayed.
sbrCaps	1	Caps Lock status displayed.
sbrNum	2	Number Lock status displayed.
sbrIns	3	Insert key status displayed.
sbrScrl	4	Scroll Lock status displayed.
sbrTime	5	Time displayed in System format.
sbrDate	6	Date displayed in System format.
sbrKana	7	Kana. displays the letters KANA in bold when scroll lock is enabled, and dimmed when disabled.

Panels Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolPanelsCollectionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolPanelsCollectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vamthItem;vbcolPanelsCollectionP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vamthClear;vamthRemove;vbcolPanelsCollectionM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbcolPanelsCollectionE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcolPanelsCollectionS"}

A **Panels** collection contains a collection of **Panel** objects.

Syntax

statusbar.**Panels**(*index*)

The **Panels** collection syntax has these parts.

Part	Description
statusbar	An object expression that evaluates to a **StatusBar** control.
Index	An integer or string that uniquely identifies the object in the collection. The integer is the value of the **Index** property of the desired **Panel** object; the string is the value of the **Key** property of the desired **Panel** object.

The **Panels** collection is a 1-based array of **Panel** objects. By default, there is one **Panel** object on a **StatusBar** control. Therefore, if you want three panels to be created, you only need to add two objects to the **Panels** collection.

The **Panels** property returns a reference to a **Panels** collection.

To add a **Panel** object to a collection, use the **Add** method for **Panel** objects at run time, or the Panel Properties tab on the Status Bar Control Properties dialog box at design time.

Each item in the collection can be accessed by its **Index** property or its **Key** property. For example, to get a reference to the third **Panel** object in a collection, use the following syntax:

```
Dim pnlX As Panel
Set pnlX = StatusBar1.Panels(3)    ' Reference by index number.
   ' or
Set pnlX = StatusBar1.Panels("Third")    ' Reference by unique key.
   ' or
Set pnlX = StatusBar1.Panels.Item(3)     ' Use Item method.
```

Panel Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjPanelC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjPanelX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjPanelP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjPanelM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjPanelE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjPanelS"}

A **Panel** object represents an individual panel in the **Panels** collection of a **StatusBar** control.

Remarks

A **Panel** object can contain text and a bitmap which may be used to reflect the status of an application.

Use the **Panels** collection to retrieve, add, or remove an individual **Panel** object.

To change the look of a panel, change the properties of the **Panel** object. To modify the properties at design-time, you can change the properties of the **Panel** object in the Properties Page. At run-time, you can change the **Panel** object properties in code.

Add Method (Panels Collection)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthAddPanelsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAddPanelsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAddPanelsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAddPanelsS"}

Adds a **Panel** object to a **Panels** collection and returns a reference to the newly created **Panel** object. Doesn't support named arguments.

Syntax

object.**Add***(index, key, text, style, picture)*

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

Part	Description
object	An object expression that evaluates to a **Panels** collection.
index	Optional. An integer specifying the position where the **Panel** object is to be inserted. If no *index* is specified, the **Panel** is added to the end of the **Panels** collection.
key	Optional. A unique string that identifies the **Panel**. Use *key* to retrieve a specific **Panel**. This is equivalent to setting the **Key** property of the new **Panel** object after the object has been added.
text	Optional. A string that appears in the **Panel**. This is equivalent to setting the **Text** property of the new **Panel** object after the object has been added.
style	Optional. The style of the panel. The available styles are detailed in the **Style** Property (**Panel** Object). This is equivalent to setting the **Style** property of the new **Panel** object after the object has been added.
picture	Optional. Specifies the bitmap displayed in the active **Panel**. For more information, see the **LoadPicture** function. This is equivalent to setting the **Picture** property of the new **Panel** object after the object has been added.

Remarks

At run time, the **Add** method returns a reference to the newly inserted **Panel** object. With this reference, you can set properties for every new **Panel** in the following manner:

```
Dim pnlX As Panel
Dim i As Integer
For i = 1 To 6  ' Add six Panel objects.
    ' Create a panel and get a reference to it simultaneously.
    Set pnlX = StatusBar1.Panels.Add(, "Panel" & i) ' Set Key property.
    pnlX.Style = i  ' Set Style property.
    pnlX.AutoSize = sbrContents ' Set AutoSize property.
Next i
```

The value of the **Text** property is displayed in a **Panel** object when the **Panel** object's **Style** property is set to **sbrText**.

The **Panels** collection is a 1-based collection. In order to get a reference to the first (default) **Panel** in a collection, you can use its **Index** or **Key** (if there is one) properties, or the **Item** method. The following code references the first **Panel** object using its index.

```
Dim pnlX As Panel
' Get a reference to first Panel.
Set pnlX = StatusBar1.Panels(1) ' Use the index
pnlX.Text = "Changed text" ' Alter the Panel object's text.
```

By default, one **Panel** already exists on the control. Therefore, after adding panels to a collection, the **Count** will be one more than the number of panels added. For example:

```
Dim i as Integer
For i = 1 to 4  ' Add four panels.
   StatusBar1.Panels.Add   ' Add panels without any properties.
Next i
MsgBox StatusBar1.Panels.Count   ' Returns 5 panels.
```

Add Method (Panels Collection) Example

This example uses the **Add** method to add three new **Panel** objects to a **StatusBar** control. To use the example, place a **StatusBar** control on a form and paste the code into the form's Declarations section. Run the example.

```
Private Sub Form_Load()
Dim pnlX As Panel
    ' Add blank panel as a spacer
    Set pnlX = StatusBar1.Panels.Add()
    pnlX.AutoSize = sbrSpring
    pnlX.MinWidth = 1
    ' Add a panel with a clock icon and time style.
    Set pnlX = StatusBar1.Panels.Add _
    (, , , sbrTime, LoadPicture("\icons\misc\clock03.ico"))
    ' Add second panel, with bitmap and Date style.
    Set pnlX = StatusBar1.Panels.Add _
    (, , , sbrDate, LoadPicture("\bitmaps\assorted\calendar.bmp"))
    ' Set Bevel property for last Panel object.
    pnlX.Bevel = sbrInset   ' Inset bevel.
    pnlX.Alignment = sbrRight   ' Set Alignment property for last object.
    ' Set Text and AutoSize properties for first (default )Panel object.
    StatusBar1.Panels(1).Text = "Add Panel Example"
    StatusBar1.Panels(1).AutoSize = sbrContents
End Sub
```

Alignment Property (Panel Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAlignmentPC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAlignmentPanelX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAlignmentPanelA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAlignmentPanelS"}

Returns or sets the alignment of text in the caption of a **Panel** object in a **StatusBar** control.

Syntax

*object.***Alignment** [= *number*]

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

Part	Description
object	An object expression that evaluates to a **Panel** object.
number	A constant or value specifying the type of action, as described in Settings.

Settings

The settings for *number* are:

Constant	Value	Description
sbrLeft	0	(Default). Text appears left-justified and to right of bitmap.
sbrCenter	1	Text appears centered and to right of bitmap.
sbrRight	2	Text appears right-justified and to left of bitmap.

Remarks

As well as positioning the text, the **Alignment** property specifies the position of the bitmap, as described in Settings. There is no way to independently position the bitmap within the panel.

Alignment Property (Panel Object) Example

This example adds two **Panel** objects to a **StatusBar** control and aligns the text in each panel using one of the three available styles. To try the example, place a **StatusBar** control on a form and paste the code into the Declarations section of the form. Run the example.

```
Sub Form_Load()
    ' Declare variables.
    Dim pnlX As Panel
    Dim I As Integer

    For I = 1 To 2  ' Add two panels.
      StatusBar1.Panels.Add
    Next I

    For I = 1 To 3  ' Add pictures to each Panel.
      Set pnlX = StatusBar1.Panels(I)
      Set pnlX.Picture = LoadPicture("\icons\comm\net12.ico")
      ' Set AutoSize and MinWidth so that panels
      ' are always in view.
      pnlX.AutoSize = sbrSpring
      pnlX.MinWidth = 1
    Next I

    ' Set styles and alignment.
    With StatusBar1.Panels
      .Item(1).Text = "Left"
      .Item(1).Alignment = sbrLeft ' Left alignment.
      .Item(2).Text = "Center"
      .Item(2).Alignment = sbrCenter ' Centered alignment.
      .Item(3).Text = "Right"
      .Item(3).Alignment = sbrRight ' Right alignment.
    End With
End Sub
```

AutoSize Property (Panel Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAutoSizePC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAutoSizePX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAutoSizePA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAutoSizePS"}

Returns or sets a value that determines the width of a **Panel** object after the **StatusBar** control has been resized.

Syntax

object.**AutoSize** [= *number*]

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

Part	Description
object	An object expression that evaluates to a **Panel** object.
number	A constant or value specifying the type of action, as described in Settings.

Settings

The settings for *number* are:

Constant	Value	Description
sbrNoAutoSize	0	(Default) None. No autosizing occurs. The width of the **Panel** is always and exactly that specified by the **Width** property.
sbrSpring	1	Spring. When the parent form resizes and there is extra space available, all panels with this setting divide the space and grow accordingly. However, the panels' width never falls below that specified by the **MinWidth** property.
sbrContents	2	Content. The **Panel** is resized to fit its contents, however, the width will never fall below the width specified by the **MinWidth** property.

Remarks

Panel objects with the Contents style have precedence over those with the Spring style. This means that a Spring-style **Panel** is shortened if a **Panel** with the Contents style requires that space.

AutoSize Property (Panel Object) Example

This example adds two **Panel** objects to a **StatusBar** control and sets the **AutoSize** property to Content for all panels. As the cursor is moved over the objects on the form, the x and y coordinates are displayed as well as the **Tag** property value for each control. To try the example, place a **StatusBar**, a **PictureBox**, and a **CommandButton** on a form, then paste the code into the Declarations section. Run the example and move the cursor over the various controls.

```
Private Sub Form_Load()
   Dim pnlX As Panel
   ' Set long tags for each object.
   Form1.Tag = "Project 1 Form"
   Command1.Tag = "A command button"
   Picture1.Tag = "Picture Box Caption"
   StatusBar1.Tag = "Application StatusBar1"
   ' Set the AutoSize style of the first panel to Contents.
   StatusBar1.Panels(1).AutoSize = sbrContents
   ' Add 2 more panels, and set them to Contents.
   Set pnlX = StatusBar1.Panels.Add
   pnlX.AutoSize = sbrContents
   Set pnlX = StatusBar1.Panels.Add
   pnlX.AutoSize = sbrContents
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, x As
Single, y As Single)
   ' Display the control's tag in panel 1, and x and y
   ' coordinates in panels 2 and 3. Because AutoSize = Contents,
   ' the first panel stretches to accommodate the varying text.
   StatusBar1.Panels(1).Text = Form1.Tag
   StatusBar1.Panels(2).Text = "X = " & x
   StatusBar1.Panels(3).Text = "Y = " & y
End Sub

Private Sub Command1_MouseMove(Button As Integer, Shift As Integer, x As
Single, y As Single)
   StatusBar1.Panels(1).Text = Command1.Tag
   StatusBar1.Panels(2).Text = "X = " & x
   StatusBar1.Panels(3).Text = "Y = " & y
End Sub

Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, x As
Single, y As Single)
   StatusBar1.Panels(1).Text = Picture1.Tag
   StatusBar1.Panels(2).Text = "X = " & x
   StatusBar1.Panels(3).Text = "Y = " & y
End Sub

Private Sub StatusBar1_MouseMove(Button As Integer, Shift As Integer, x As
Single, y As Single)
   StatusBar1.Panels(1).Text = StatusBar1.Tag
   StatusBar1.Panels(2).Text = "X = " & x
   StatusBar1.Panels(3).Text = "Y = " & y
End Sub
```

Bevel Property (Panel Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBevelC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBevelX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBevelA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBevelS"}

Returns or sets the bevel style of a **StatusBar** control's **Panel** object.

Syntax

*object.***Bevel** [= *value*]

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

Part	Description
object	An object expression that evaluates to a **Panel** object.
value	A constant or value which determines the bevel style, as specified in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
sbrNoBevel	0	None. The **Panel** displays no bevel, and text looks like it is displayed right on the status bar.
sbrInset	1	(Default). Inset. The **Panel** appears to be sunk into the status bar.
sbrRaised	2	Raised. The **Panel** appears to be raised above the status bar.

Bevel Property (Panel Object) Example

This example adds two **Panel** objects to a **StatusBar** control, and gives each **Panel** a different bevel style. To use the example, place a **StatusBar** control on a form and paste the code into the Declarations section. Run the example.

```
Private Sub Form_Load()
   Dim pnlX As Panel
   Dim I as Integer

   For I = 1 to 2
      Set pnlX = StatusBar1.Panels.Add() ' Add 2 panels.
   Next I

   With StatusBar1.Panels
      .Item(1).Style = sbrCaps     ' Caps Lock
      .Item(1).Bevel = sbrInset    ' Inset
      .Item(2).Style = sbrNum      ' NumLock
      .Item(2).Bevel = sbrNoBevel  ' No bevel
      .Item(3).Style = sbrDate     ' Date
      .Item(3).Bevel = sbrRaised   ' Raised bevel
   End With
End Sub
```

MinWidth Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMinWidthC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMinWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMinWidthA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMinWidthS"}

Returns or sets the minimum width of a **StatusBar** control's **Panel** object.

Syntax

*object.***MinWidth** [= *value*]

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

Part	Description
object	An object expression that evaluates to a **Panel** object.
value	An integer that determines the minimum width of a **Panel** object. The scale mode for this value is determined by the container of the control.

Remarks

The **MinWidth** property is used when the **AutoSize** property is set to Contents or Spring, to prevent the panel from autosizing to a width that is too small. When the **AutoSize** property is set to None, the **MinWidth** property is always set to the same value as the **Width** property.

The default value is the same as the default of the **Width** property. The *value* argument uses the same scale units as the scale mode of the parent form or container.

MinWidth Property Example

This example uses the default panel of a **StatusBar** control to display the current date. The **MinWidth** property is set so that when you click on the panel, the date is cleared but the panel remains the same size. To use the example, place a **StatusBar** control on a form, and paste the code into the Declarations section. Run the example and click on the **Panel** object to clear the date.

```
Private Sub Form_Load()
    StatusBar1.Panels(1).AutoSize = sbrContents
    StatusBar1.Panels(1).Text = "Today's Date is: " & Str(Now)
    ' Set minimum width to the current size of panel
    StatusBar1.Panels(1).MinWidth = StatusBar1.Panels(1).Width
End Sub

Private Sub StatusBar1_PanelClick(ByVal Panel As ComctlLib.Panel)
    ' Clear today's date but keep size at minimum width.
    Panel.Text = "Today's Date is: "
End Sub
```

Panels Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPanelsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPanelsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPanelsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPanelsS"}

Returns a reference to a collection of **Panel** objects.

Syntax

object.**Panels**

The *object* placeholder is an object expression that evaluates to a **StatusBar** control.

PanelClick Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtPanelClickC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtPanelClickX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtPanelClickA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtPanelClickS"}

The PanelClick event is similar to the standard Click event but occurs when a user presses and then releases a mouse button over any of the **StatusBar** control's **Panel** objects.

Syntax

Private Sub *object*_**PanelClick(ByVal** *panel* **As Panel)**

The PanelClick event syntax has these parts:

Part	Description
object	An object expression that evaluates to a **StatusBar** control.
panel	A reference to a **Panel** object.

Remarks

The standard Click event also occurs when a **Panel** object is clicked.

The PanelClick event is only generated when the click occurs over a **Panel** object. When the **StatusBar** control's **Style** property is set to Simple style, panels are hidden, and therefore the PanelClick event is not generated.

You can use the reference to the **Panel** object to set properties for that panel. For example, the following code resets the **Bevel** property of a clicked **Panel**:

```
Private Sub StatusBar1_PanelClick(ByVal Panel As Panel)
   Select Case Panel.Key
   Case "DisplayFileName" ' Key="DisplayFileName"
     Panel.Bevel = sbrRaised ' Reset Bevel property
   ' Add other case statements for other panels
   End Select
End Sub
```

PanelClick Event Example

This example adds two **Panel** objects to a **StatusBar** control; when each **Panel** is clicked, the value of the **Key** and **Width** properties of the clicked **Panel** are displayed in the third **Panel**. To try the example, place a **StatusBar** control on a form and paste the code into the Declarations section. Run the example.

```
Private Sub Form_Load()
   Dim I as Integer
   For I = 1 to 2
      StatusBar1.Panels.Add
   Next I

   With StatusBar1.Panels
      .Item(1).Style = sbrDate
      .Item(1).Key = "Date panel"
      .Item(1).AutoSize = sbrContents
      .Item(1).MinWidth = 2000
      .Item(2).Style = sbrTime
      .Item(2).Key = "Time panel"
      .Item(3).AutoSize = sbrContents    ' Content
      .Item(3).Text = "Panel 3"
      .Item(3).Key = "Panel 3"
   End With
End Sub

Private Sub StatusBar1_PanelClick(ByVal Panel As Panel)
   ' Show clicked panel's key and width in Panel 3.
   StatusBar1.Panels("Panel 3").Text = Panel.Key & " Width = " &
Panel.Width
End Sub
```

PanelDblClick Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtPanelDblClickC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtPanelDblClickX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtPanelDblClickA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtPanelDblClickS"}

The PanelDblClick event is similar to the standard DblClick Event but occurs when a user presses and then releases a mouse button twice over a **StatusBar** control's **Panel** object.

Syntax

Sub *object*_**PanelDblClick(ByVal** *panel* **As Panel)**

The PanelDblClick event syntax has these parts:

Part	Description
object	An object expression that evaluates to a **StatusBar** control.
panel	A reference to the double-clicked **Panel**.

Remarks

The standard DblClick event also occurs when a **Panel** is double-clicked.

The PanelDblClick event is only generated when the double-click occurs over a **Panel** object. When the **StatusBar** control's **Style** property is set to Simple style, panels are hidden, and therefore the PanelDblClick event is not generated.

PanelDblClick Event Example

This example adds two **Panel** objects to a **StatusBar** control. When the user double-clicks on the control, the text of the clicked **Panel** object is displayed. To try the example, place a **StatusBar** control on a form and paste the code into the form's Declarations section. Run the example and double-click on the control.

```
Private Sub Form_Load()
Dim I as Integer
   For I = 1 to 2
      StatusBar1.Panels.Add
   Next I

   With StatusBar1.Panels
      .Item(1).Text = "A long piece of information."
      .Item(1).AutoSize = sbrContents    ' Content
      .Item(2).Style = sbrDate     ' Date style
      .Item(2).AutoSize = sbrContents    ' Content
      .Item(3).Style = sbrTime     ' Time style
   End With
End Sub

Private Sub StatusBar1_PanelDblClick(ByVal Panel As Panel)
   MsgBox "Panel.Style = " & Panel.Style
End Sub
```

SimpleText Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSimpleTextC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSimpleTextX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSimpleTextA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSimpleTextS"}

Returns or sets the text displayed when a **StatusBar** control's **Style** property is set to Simple.

Syntax

object.**SimpleText** [= *string*]

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

Part	Description
object	An object expression that evaluates to a **StatusBar** control.
string	A string that is displayed when the **Style** property is set to Simple.

Remarks

The **StatusBar** control has a **Style** property which can be toggled between Simple and Normal styles. When in Simple style, the status bar displays only one panel. The text displayed in Simple style is also different from that displayed in Normal style. This text is set with the **SimpleText** property.

The **SimpleText** property can be used in situations where an application's mode of operation temporarily switches. For example, when a menu is pulled down, the **SimpleText** could describe the menu's purpose.

SimpleText Property Example

This example adds two **Panel** objects to a **StatusBar** control that appear in Normal style, and then adds a string (using the **SimpleText** property) that appears when the **Style** property is set to Simple. The control toggles between the Simple style and the Normal style. To try the example, place a **StatusBar** control on a form and paste the code into the Declarations section of the form. Run the example and click on the **StatusBar** control.

```
Private Sub Form_Load()
   Dim I As Integer
   For I = 1 to 2
      StatusBar1.Panels.Add  ' Add 2 Panel objects.
   Next I

   With StatusBar1.Panels
      .Item(1).Style = sbrNum      ' Number lock
      .Item(2).Style = sbrCaps     ' Caps lock
      .Item(3).Style = sbrScrl     ' Scroll lock
   End With
End Sub

Private Sub StatusBar1_Click()
   ' Toggle between simple and normal style.
   With StatusBar1
      If .Style = 0 Then
         ' This text will be displayed when the StatusBar is in Simple
style.
         .SimpleText = "Date and Time: " & Now
         .Style = sbrSimple  ' Simple style.
      Else
         .Style = sbrNormal  ' Normal style.
      End If
   End With
End Sub
```

Style Property (StatusBar Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproStatusStyleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproStyleStatusbarX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStyleStatusbarA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStyleStatusbarS"}

Returns or sets the style of a **StatusBar** control.

Syntax

object.**Style** [= *number*]

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

Part	Description
object	An object expression that evaluates to a **StatusBar** control.
number	An integer or constant that determines the appearance of the **StatusBar** control, as specified in Settings.

Settings

The settings for *number* are:

Constant	Value	Description
sbrNormal	0	(Default). Normal. The **StatusBar** control shows all **Panel** objects.
sbrSimple	1	Simple. The control displays only one large panel.

Remarks

The **StatusBar** can toggle between two modes: Normal and Simple. When in Simple style, the **StatusBar** displays only one panel. The appearance also changes: the bevel style is raised with no borders. This allows the control to have two appearances, both of which are maintained separately from each other.

You can display different strings depending on the control's style. Use the **SimpleText** property to set the text of the string to be displayed when the **Style** property is set to Simple.

Note When the **Style** property is set to Simple, the **StatusBar** control displays a large panel (the width of the control) which cannot be controlled through the Panels collection.

Style Property (StatusBar Control) Example

This example adds two **Panel** objects to a **StatusBar** control that appear in Normal style, and then adds a string (using the **SimpleText** property) that will appear when the **Style** property is set to Simple. The control toggles between the Simple style and the Normal style to show the **SimpleText** property string. To try the example, place a **StatusBar** control on a form and paste the code into the Declarations section of the form. Run the example and click on the **StatusBar** control.

```
Private Sub Form_Load()
   Dim I As Integer
   For I = 1 to 2
      StatusBar1.Panels.Add
   Next I
   With StatusBar1.Panels
      .Item(1).Style = sbrDate     ' Date
      .Item(2).Style = sbrCaps     ' Caps lock
      .Item(3).Style = sbrScrl     ' Scroll lock
   End With

End Sub

Private Sub StatusBar1_Click()
   With StatusBar1
      If .Style = sbrNormal Then
         .SimpleText = Time  ' Show the time.
         .Style = sbrSimple  ' Simple style
      Else
         .Style = sbrNormal  ' Normal style
      End If
   End With
End Sub
```

Style Property (Panel Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPanelStyleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproStylePanelX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStylePanelA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStylePanelS"}

Returns or sets the style of a **StatusBar** control's **Panel** object.

Syntax

object.**Style** [= *number*]

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

Part	Description
object	An object expression that evaluates to a **Panel** object.
number	An integer or constant specifying the style of the **Panel**, as described in Settings.

Settings

The settings for *number* are:

Constant	Value	Description
sbrText	0	(Default). Text and/or a bitmap. Set text with the **Text** property.
sbrCaps	1	Caps Lock key. Displays the letters CAPS in bold when Caps Lock is enabled, and dimmed when disabled.
sbrNum	2	Number Lock. Displays the letters NUM in bold when the number lock key is enabled, and dimmed when disabled.
sbrIns	3	Insert key. Displays the letters INS in bold when the insert key is enabled, and dimmed when disabled.
sbrScrl	4	Scroll Lock key. Displays the letters SCRL in bold when scroll lock is enabled, and dimmed when disabled.
sbrTime	5	Time. Displays the current time in the system format.
sbrDate	6	Date. Displays the current date in the system format.
sbrKana	7	Kana. displays the letters KANA in bold when scroll lock is enabled, and dimmed when disabled.

Remarks

If you set the **Style** property to any style except 0 (text and bitmap), any text set with the **Text** property will not display unless the **Style** property is set to 0.

The **Style** property can be set as **Panel** objects are added to a collection. See the **Add** method for more information.

Note The **StatusBar** control also has a **Style** property. When the **StatusBar** control's **Style** is set to Simple, the control displays only one large panel and its string (set with the **SimpleText** property).

Style Property (Panel Object) Example

This example displays data in the various styles on a **StatusBar** control. To try this example, place a **StatusBar** control on a form and paste the code into the form's Declarations section, and run the example.

```
Private Sub Form_Load()
   ' Dim variables.
   Dim I as Integer
   Dim pnlX as Panel

   For I = 1 to 5 ' Add 5 panels.
      Set pnlX = StatusBar1.Panels.Add( )
   Next I

   ' Set the style of each panel.
   With StatusBar1.Panels
      .Item(1).Style = sbrDate     ' Date
      .Item(2).Style = sbrTime     ' Time
      .Item(3).Style = sbrCaps     ' Caps lock
      .Item(4).Style = sbrNum      ' Number lock
      .Item(5).Style = sbrIns      ' Insert key
      .Item(6).Style = sbrScrl     ' Scroll lock
   End With
   Form1.Width = 9140 ' Widen form to show all panels.
End Sub
```

Width Property (Panel Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPanelWidthC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPanelWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPanelWidthA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPanelWidthS"}

Returns or sets the current width of a **StatusBar** control's **Panel** object.

Syntax

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

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

Part	Description
object	An object expression that evaluates to a **Panel** object.
number	An integer that determines the width of the **Panel**.

Remarks

The **Width** property always reflects the actual width of a **Panel** and can't be smaller than the **MinWidth** property.

Width Property (Panel Object) Example

This example creates three **Panel** objects and sets their **Width** property to different values. When you click on the form, the **Width** property of the first **Panel** is reset. 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 each panel to see its width.

```
Private Sub Form_Load()
   Dim X As Panel
   Dim I as Integer
   For I = 1 to 2 ' Add 2 panels.
      Set X = StatusBar1.Panels.Add()
   Next I
   With StatusBar1.Panels
      .Item(1).Text = "Path = " & App.Path
      .Item(1).AutoSize = sbrContents    ' Contents
      .Item(1).Width = 2000  ' A long panel
      .Item(2).Text = "Record Field"
      .Item(2).AutoSize = sbrSpring      ' Spring
      .Item(2).Width = 1000  ' A medium panel
      .Item(3).Style = sbrTime     ' Time
      .Item(3).AutoSize = sbrSpring      ' Spring
      .Item(3).Width = 500 ' A medium panel
   End With
End Sub

Private Sub Form_Click()
   ' Change Width.
   StatusBar1.Panels(1). Width = 800
End Sub
```

TabStrip Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjTabControlC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjTabControlX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjTabControlP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjTabControlM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjTabControlE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjTabControlS"}

A **TabStrip** control is like the dividers in a notebook or the labels on a group of file folders. By using a **TabStrip** control, you can define multiple pages for the same area of a window or dialog box in your application.

Syntax

TabStrip

Remarks

The control consists of one or more **Tab** objects in a **Tabs** collection. At both design time and run time, you can affect the **Tab** object's appearance by setting properties. You can also add and remove tabs in the properties dialog box at design time, or add and remove **Tab** objects at run time using methods.

The **Style** property determines whether the **TabStrip** control looks like push buttons (Buttons) or notebook tabs (Tabs). At design time when you put a **TabStrip** control on a form, it has one notebook tab. If the **Style** property is set to **tabTabs**, then there will be a border around the **TabStrip** control's internal area. When the **Style** property is set to **tabButtons**, no border is displayed around the internal area of the control, however, that area still exists.

To set the overall size of the **TabStrip** control, use its drag handles and/or set the **Top**, **Left**, **Height**, and **Width** properties. Based on the control's overall size at run time, Visual Basic automatically determines the size and position of the internal area and returns the Client-coordinate properties – **ClientLeft**, **ClientTop**, **ClientHeight**, and **ClientWidth**. The **MultiRow** property determines whether the control can have more than one row of tabs, the **TabWidthStyle** property determines the appearance of each row, and, if **TabWidthStyle** is set to **tabFixed**, you can use the **TabFixedHeight** and **TabFixedWidth** properties to set the same height and width for all tabs in the **TabStrip** control.

The **TabStrip** control is not a container. To contain the actual pages and their objects, you must use **Frame** controls or other containers that match the size of the internal area which is shared by all **Tab** objects in the control. If you use a control array for the container, you can associate each item in the array with a specific **Tab** object, as in the following example:

```
' This code makes the selected tab's frame container
' the topmost in the ZOrder
Frame1(TabStrip1.SelectedItem.Index - 1).ZOrder 0
```

Tip Use a **Frame** control with its **BorderStyle** set to None as the container instead of a **PictureBox** control. A **Frame** control uses less overhead than a **PictureBox** control.

The **Tabs** property of the **TabStrip** control is the collection of all the **Tab** objects. Each **Tab** object has properties associated with its current state and appearance. For example, you can associate an **ImageList** control with the **TabStrip** control, and then use images on individual tabs. You can also associate a ToolTip with each **Tab** object.

Distribution Note The **TabStrip** control is part of a group of custom controls that are found in the COMCTL32.OCX file. To use the **TabStrip** control in your application, you must add the COMCTL32.OCX file to the project. When distributing your application, install the COMCTL32.OCX file in the user's Microsoft Windows SYSTEM directory. For more information on how to add a custom control to a project, see the *Programmer's Guide*.

Tab Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjTabC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjTabX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjTabP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjTabM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjTabE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjTabS"}

A **Tab** object represents an individual tab in the **Tabs** collection of a **TabStrip** control.

Remarks

For each **Tab** object, you can use various properties to specify its appearance, and you can specify its state with the **Selected** property.

At design time, use the Insert Tab and Remove Tab buttons on the Tabs tab in the TabStrip Control Properties dialog box to insert and remove tabs, and use the text boxes to specify any of these properties for a **Tab** object: **Caption**, **Image**, **ToolTipText**, **Tag**, **Index**, and/or **Key**. You can also specify these properties at run time.

Use the **Caption** and **Image** properties, separately or together, to label or put an icon on a tab.

- To use the **Caption** property, in the Caption text box on the Tabs tab in the TabStrip Control Properties dialog box, type the text you want to appear on the tab or button at run time.
- To use the **Image** property, put an **ImageList** control on the form and fill the **ListImages** collection with **ListImage** objects, each of which has an index number and an optional key, if you add one. On the General tab in the **TabStrip** Control Properties dialog box, select that **ImageList** to associate it with the **TabStrip** control. In the Image text box on the Tabs tab, type the index number or key of the **ListImage** object that should appear on the **Tab** object.

Use the **ToolTipText** property to temporarily display a string of text in a small rectangular box at run time when the user's cursor hovers over the tab. To set the **ToolTipText** property at design time, select the **ShowTips** checkbox on the General tab, and then in the ToolTipText text box on the Tabs tab, type the ToolTip string.

To return a reference to a **Tab** object a user has selected, use the **SelectedItem** property; to determine whether a specific tab is selected, use the **Selected** property. These properties are useful in conjunction with the BeforeClick event to verify or record data associated with the currently-selected tab before displaying the next tab the user selects.

Each **Tab** object also has read-only properties you can use to reference a single **Tab** object in the **Tabs** collection: **Left**, **Top**, **Height** and **Width**.

Tabs Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolTabsCollectionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolTabsCollectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vamthItem;vbcolTabsCollectionP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vamthClear;vamthRemove;vbcolTabsCollectionM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbcolTabsCollectionE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcolTabsCollectionS"}

A **Tabs** collection contains a collection of **Tab** objects.

Syntax

tabstrip.**Tabs**(*index*)
tabstrip.**Tabs**.**Item**(*index*)

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

Part	Description
tabstrip	An object expression that evaluates to a **TabStrip** control.
index	An integer or string that uniquely identifies a member of an object collection. The integer is the value of the **Index** property of the desired **Tab** object; the string is the value of the **Key** property of the desired **Tab** object.

At design time, use the Insert Tab and Remove Tab buttons on the Tabs tab in the TabStrip Control Properties dialog box to add and remove **Tab** objects from the **Tabs** collection.

The **Tabs** collection uses the **Count** property to return the number of tabs in the collection. To manipulate the **Tab** objects in the **Tabs** collection, use these methods at run time:

- **Add** – adds **Tab** objects to the **TabStrip** control.
- **Item** – retrieves the **Tab** identified by its **Key** or **Index** from the collection.
- **Clear** – removes all **Tab** objects from the collection.
- **Remove** – removes the **Tab** identified by its **Key** or **Index** from the collection.

TabStrip Control Constants

Tab Style Constants

Constant	Value	Description
tabTabs	0	Tabs appear as notebook tabs, and the internal area has a three-dimensional border enclosing it.
tabButtons	1	Tabs appear as push buttons, and the internal area has no border around it.

Tab Width Style Constants

Constant	Value	Description
tabJustified	0	Each tab is wide enough to accommodate its contents, and the width of each tab is increased, if needed, so that each row of tabs spans the width of the control. If there is only a single row of tabs, this style has no effect.
tabNonJustified	1	Each tab is just wide enough to accommodate its contents. The rows are not justified, so multiple rows of tabs are jagged.
tabFixed	2	The height and width of all tabs are identical, and are set by the **TabFixedHeight** and **TabFixedWidth** properties.

Add Method (Tabs Collection)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthAddTabsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAddTabsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAddTabsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAddTabsS"}

Adds a **Tab** object to a **Tabs** collection in a **TabStrip** control. Doesn't support named arguments.

Syntax

object.**Add(***index, key, caption, image***)**

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

Part	Description
object	An object expression that evaluates to a **Tabs** collection.
index	Optional. An integer specifying the position where you want to insert the **Tab**. If you don't specify an index, the **Tab** is added to the end of the **Tabs** collection.
key	Optional. A unique string that identifies the **Tab**. Use key to retrieve a specific Tab. This is equivalent to setting the **Key** property of the new **Tab** object after the object has been added to the **Tabs** collection.
caption	Optional. The string that appears on the **Tab**. This is equivalent to setting the **Caption** property of the new **Tab** object after the object has been added to the **Tabs** collection.
image	Optional. The index of an image in an associated **ImageList** control. This image is displayed on the tab. This is equivalent to setting the **Image** property of the new **Tab** object after the object has been added to the **Tabs** collection.

Remarks

To add tabs to the **TabStrip** control at design time, click the Insert Tab button on the Tab tab in the TabStrip Control Properties dialog box, and then fill in the appropriate fields for the new tab.

To add tabs to the **TabStrip** control at run time, use the **Add** method, which returns a reference to the newly inserted **Tab** object. For example, the following code adds a tab with the *caption*, "Howdy!" whose *key* is "MyTab," as the second tab (its *index* is 2):

```
Set X = TabStrip1.Tabs.Add(2,"MyTab","Howdy!")
```

Add Method (Tabs Collection) Example

This example adds three **Tab** objects, each with captions and images from an **ImageList** control, to a **TabStrip** control. To try this example, put an **ImageList** and a **TabStrip** control on a form. The **ImageList** control supplies the images for the **Tab** objects, so add three images to the **ImageList** control. Paste the following code into the Load event of the Form object, and run the program.

```
Private Sub Form_Load()
   Dim X As Integer
   Set TabStrip1.ImageList = ImageList1
   TabStrip1.Tabs(1).Caption = "Time"
   TabStrip1.Tabs.Add 2, , "Date"
   TabStrip1.Tabs.Add 3, , "Mail"
   For X = 1 To TabStrip1.Tabs.Count
      TabStrip1.Tabs(X).Image = X
   Next X
End Sub
```

BeforeClick Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vafctInputBox;vafctMsgBox;vbevtBeforeClickC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtBeforeClickX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtBeforeClickA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtBeforeClickS"}

Generated when a **Tab** object in a **TabStrip** control is clicked, or a **Tab** object's **Selected** setting has changed.

Syntax

Private Sub *object*_**BeforeClick(***cancel* **As Integer)**

The BeforeClick event syntax has these parts:

Part	Description
object	An object expression that evaluates to a **TabStrip** control.
cancel	Evaluates to an integer with values of 0 (False) and -1 (True). The initial value is 0.

Remarks

Use the BeforeClick event to validate the information on the old **Tab** object before actually generating a Click event that selects the new **Tab** object. Setting the *cancel* argument to **True** allows you to stop a change to the new selection.

Note Setting the *cancel* argument to **True** prevents the focus from switching to another tab but doesn't stop the Click event from occurring.

Note If you use the **MsgBox** or **InputBox** functions during the BeforeClick event procedure, the **TabStrip** control will not receive a Click event, regardless of the setting of the *cancel* argument.

BeforeClick Event Example

This example uses the BeforeClick event to demonstrate how to prevent a user from switching to another tab. This is useful when you want to verify information on the current tab before displaying the newly selected tab.

To try this example, place a **TabStrip** control and a two-element **Frame** control array on the form (set the BorderStyle properties to None). In the first **Frame** control, add a **CheckBox** control and in the second, add a **TextBox**. Paste the following code into the Load event of the Form object, and run the program. Click the tab labeled Text after you select/deselect the CheckBox on the tab labeled Check.

```
Private Sub Form_Load()
Dim i As Integer
Dim Tabx As Object
' Sets the caption of the first tab to "Check."
TabStrip1.Tabs(1).Caption = "Check"
' Adds a second tab with "Text" as its caption.
Set Tabx = TabStrip1.Tabs.Add(2, , "Text")
' Labels the checkbox.
Check1.Caption = "Cancel tab switch"
   ' Aligns the Frames with the internal area
   ' of the Tabstrip Control.
   For i = 0 To 1
      Frame1(i).Left = TabStrip1.ClientLeft
      Frame1(i).Top = TabStrip1.ClientTop
      Frame1(i).Height = TabStrip1.ClientHeight
      Frame1(i).Width = TabStrip1.ClientWidth
   Next
   ' Puts the first tab's Frame container on top.
   Frame1(0).ZOrder 0
End Sub

' The BeforeClick event verifies the check box value
' to determine whether to proceed with the Click event.
Private Sub TabStrip1_BeforeClick(Cancel As Integer)
   If TabStrip1.Tabs(1).Selected Then
      If Check1.Value = 1 Then Cancel = True
   End If
End Sub

Private Sub TabStrip1_Click()
   Frame1(TabStrip1.SelectedItem.Index-1).ZOrder 0
End Sub
```

Caption Property (Tab Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCaptionTabObjC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCaptionTabObjX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCaptionTabObjA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vafctInputBox"}

Returns or sets the caption that appears on the tab or button of a **Tab** object in a **TabStrip** control.

Syntax

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

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

Part	Description
object	An object expression that evaluates to a **Tab** object.
string	A string expression that evaluates to the text displayed as the caption.

Remarks

You can set the **Caption** property for a **Tab** object in the **TabStrip** control at design time or at run time.

- Design time – On the Tab tab in the TabStrip Control properties dialog box, type the caption string in the Caption text box.
- Run time – Set the caption as follows:

```
TabStrip1.Tabs(1).Caption = "First Tab"
```

Or

```
TabStrip1.Tabs.Add 2, , "Second Tab"
```

Caption Property (Tab Object) Example

This example sets the **Caption** property for each of three **Tab** objects it adds to a **TabStrip** control. The caption strings are "Time," "Date," and "Mail." Each **Tab** object also displays an image from an **ImageList** control. To try this example, place an **ImageList** and a **TabStrip** control on a form. Place three sample bitmaps in the **ImageList** control. The **ImageList** control supplies the images for the **Tab** objects. Paste the following code into the Load event of the Form object, and run the program.

```
Private Sub Form_Load()
   Dim X As Integer
   ' Associate an ImageList with the TabStrip control.
   Set TabStrip1.ImageList = ImageList1
   ' Set the captions.
   TabStrip1.Tabs(1).Caption = "Time"
   TabStrip1.Tabs.Add 2, , "Date"
   TabStrip1.Tabs.Add 3, , "Mail"
   For X = 1 To TabStrip1.Tabs.Count
      ' Associate an image with a tab.
      TabStrip1.Tabs(X).Image = X
   Next X
End Sub
```

ClientHeight, ClientWidth, ClientLeft, ClientTop Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproClientHeightC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproClientHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproClientHeightA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproClientHeightS"}

Return the coordinates of the internal area (display area) of the **TabStrip** control. Read-only at run time; not available at design time.

Syntax

object.**ClientHeight**
object.**ClientWidth**
object.**ClientLeft**
object.**ClientTop**

The *object* placeholder represents an object expression that evaluates to a **TabStrip** control.

Remarks

At run time, the client-coordinate properties – **ClientLeft**, **ClientTop**, **ClientHeight**, and **ClientWidth** – automatically store the coordinates of the **TabStrip** control's internal area, which is shared by all **Tab** objects in the control. So that the controls associated with a specific **Tab** appear when that **Tab** object is selected, place the **Tab** object's controls inside a container, such as a **Frame** control, whose size and position match the client-coordinate properties. To associate a container (and its controls) with a **Tab** object, create a control array, such as a **Frame** control array.

All client-coordinate properties use the scale mode of the parent form. To place a **Frame** control so it fits perfectly in the internal area, use the following code:

```
Frame1.Left = TabStrip1.ClientLeft
Frame1.Top = TabStrip1.ClientTop
Frame1.Width = TabStrip1.ClientWidth
Frame1.Height = TabStrip1.ClientHeight
```

To create the effect of placing a new tab and its associated container on top when the tab is selected:

- Set the size and location of the container in the **TabStrip** control's internal area to the client-coordinate properties; and
- Use the **ZOrder** method to place the selected tab's container control at the front or back of the z-order.

ClientHeight, ClientWidth, ClientLeft, ClientTop Properties Example

The following example demonstrates using the Client-coordinate properties – **ClientLeft**, **ClientTop**, **ClientWidth**, and **ClientHeight** – along with a **Frame** control array to display tab – specific objects in the internal area of the **TabStrip** control when switching tabs. The example uses the **ZOrder** method to display the appropriate **Frame** control and the objects it contains.

To try this example, place a **TabStrip** control and a three-element **Frame** control array on the form. In one **Frame** control, place a **CheckBox** control, in another, place a **CommandButton** control, and in the third, place a **TextBox** control. Paste the following code into the Load event of the Form object, and run the program. Click the various tabs to select them and their contents.

```
Private Sub Form_Load()
Dim Tabx As Object
Dim i As Integer
   ' Sets the caption of the first tab to "Check."
   TabStrip1.Tabs(1).Caption = "Check"
   ' Adds a second tab with "Command" as its caption.
   Set Tabx = TabStrip1.Tabs.Add(2, , "Command")
   ' Adds a third tab with "Text" as its caption.
   Set Tabx = TabStrip1.Tabs.Add(3, , "Text")

   ' Aligns the frame containers with the internal
   ' area of the TabStrip control.
   For i = 0 To 2
      With TabStrip1
         Frame1(i).Move .ClientLeft, .ClientTop, _
         .ClientWidth, .ClientHeight
      End With
   Next
   ' Puts the first tab's picture box container on top
   ' at startup.
   Frame1(0).ZOrder 0
End Sub

Private Sub TabStrip1_Click()
   Frame1(TabStrip1.SelectedItem.Index - 1).ZOrder 0
End Sub
```

MultiRow Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMultiRowC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMultiRowX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMultiRowA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMultiRowS"}

Returns or sets a value indicating whether a **TabStrip** control can display more than one row of tabs.

Syntax

object.**MultiRow** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to a **TabStrip** control.
boolean	A boolean expression that specifies whether the control has more than one row of tabs, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	Allows more than one row of tabs.
False	Restricts tabs to a single row.

Remarks

The number of rows is automatically set by the width and number of the tabs. The number of rows can change if the control is resized, which ensures that the tab wraps to the next row. If **MultiRow** is set to **False**, and the last tab exceeds the width of the control, a horizontal spin control is added at the right end of the **TabStrip** control.

At design time, set the **MultiRow** property on the General tab in the TabStrip Properties dialog box. At run time, use code like the following to set the **MultiRow** property:

```
'Allows more than one row of tabs in the TabStrip control.
TabStrip1.MultiRow = TRUE
```

Style Property (TabStrip Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTabStyleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTabStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTabStyleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTabStyleS"}

Returns or sets the appearance – tabs or buttons – of a **TabStrip** control.

Syntax

object.**Style** [= *value*]

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

Part	Description
object	An object expression that evaluates to a **TabStrip** control.
value	A constant or integer that determines the appearance of the tabbed dialog box, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
tabTabs	0	(Default) Tabs. The tabs appear as notebook tabs, and the internal area has a three-dimensional border around it.
tabButtons	1	Buttons. The tabs appear as regular push buttons, and the internal area has no border around it.

Remarks

At design time, select the **Style** property you want – tabs or buttons – from the Style list on the General tab of the TabStrip Control Properties dialog box.

At run time, use code like the following to set the **Style** property:

```
' Style property set to the Tabs style.
TabStrip1.Style = tabTabs

' Style property set to the Buttons style:
TabStrip1.Style = tabButtons
```

Tabs Property (TabStrip Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthItem;vbproBooksOnlineJumpTopic;vbproTabsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTabsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTabsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTabsS"}

Returns a reference to the collection of **Tab** objects in a **TabStrip** control.

Syntax

object.**Tabs(***index***)**

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

Part	Description
object	An object expression that evaluates to a **TabStrip** control.
index	A value that identifies a **Tab** object in the **Tabs** collection. This may either be the **Index** property or the **Key** property of the desired **Tab** object.

Remarks

The **Tabs** collection can be accessed by using the standard collection methods, such as the **Item** method.

TabFixedHeight, TabFixedWidth Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTabFixedHeightC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTabFixedHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTabFixedHeightA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTabFixedHeightS"}

Return or set the fixed height and width of all **Tab** objects in a **TabStrip** control, but only if the **TabWidthStyle** property is set to **tabFixed**.

Syntax

object.**TabFixedHeight** [= *integer*]
object.**TabFixedWidth** [= *integer*]

The **TabFixedHeight** and **TabFixedWidth** properties syntax has these parts:

Part	Description
object	An object expression that evaluates to a **TabStrip** control.
integer	The number of pixels or twips of the height or width of a **TabStrip** control. The scale used for *integer* is dependent on the **ScaleMode** of the container.

Remarks

The **TabFixedHeight** property applies to all **Tab** objects in the **TabStrip** control. It defaults either to the height of the font as specified in the **Font** property, or the height of the **ListImage** object specified by the **Image** property, whichever is higher, plus a few extra pixels as a border. If the **TabWidthStyle** property is set to **tabFixed**, and the value of the **TabFixedWidth** property is set, the width of each **Tab** object remains the same whether you add or delete **Tab** objects in the control.

TabWidthStyle Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTabWidthStyleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTabWidthStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTabWidthStyleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTabWidthStyleS"}

Returns or sets a value that determines the justification or width of all **Tab** objects in a **TabStrip** control.

Syntax

object.**TabWidthStyle** [=*value*]

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

Part	Description
object	An object expression that evaluates to a **TabStrip** control.
value	An integer or constant that determines whether tabs are justified or set to a fixed width, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
tabJustified	0	(Default) Justified. If the **MultiRow** property is set to **True**, each tab is wide enough to accommodate its contents and, if needed, the width of each tab is increased so that each row of tabs spans the width of the control. If the **MultiRow** property is set to **False**, or if there is only a single row of tabs, this setting has no effect.
tabNonJustified	1	Nonjustified. Each tab is just wide enough to accommodate its contents. The rows are not justified, so multiple rows of tabs are jagged.
tabFixed	2	Fixed. All tabs have an identical width, which is determined by the **TabFixedWidth** property.

Remarks

At design time you can set the **TabWidthStyle** property on the General tab of the **TabStrip** Control Properties dialog box. The setting of the **TabWidthStyle** property affects how wide each **Tab** object appears at run time.

At run time, you can set the **TabWidthStyle** property as follows:

```
' Justifies all the tabs in a row to fit the width of the control.
TabStrip1.MultiRow = True
TabStrip1.TabWidthStyle = tabJustified

' Creates ragged rows of tabs.
TabStrip1.MultiRow = True
```

```
TabStrip1.TabWidthStyle = tabNonJustified

' Sets the same width for all tabs.
TabStrip1.TabFixedWidth = 500
TabStrip1.TabWidthStyle = tabFixed
```

Toolbar Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjToolbarC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjToolbarX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjToolbarP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjToolbarM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjToolbarE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjToolbarS"}

A **Toolbar** control contains a collection of **Button** objects used to create a toolbar that is associated with an application.

Syntax

Toolbar

Remarks

Typically, a toolbar contains buttons that correspond to items in an application's menu, providing a graphic interface for the user to access an application's most frequently used functions and commands.

The **Toolbar** control allows you to create toolbars by adding **Button** objects to a **Buttons** collection. Each **Button** object can have optional text or an image, or both, supplied by an associated **ImageList** control. You can display an image on a button with the **Image** property, or display text with the **Caption** property, or both, for each **Button** object. At design time, you can add **Button** objects to the control with the Toolbar Control Properties dialog box. At run time, you can add or remove buttons from the **Buttons** collection using the **Add** and **Remove** methods.

To program the **Toolbar**, add code to the ButtonClick event to respond to the selected button. You can also determine the behavior and appearance of each **Button** object using the **Style** property. For example, if four buttons are assigned the ButtonGroup style, only one button can be pressed at any time and at least one button is always pressed.

You can create space for other controls on the toolbar by assigning a **Button** object the PlaceHolder style, then positioning a control over the placeholder. For example, to place a drop-down combo box on a toolbar at design time, add a **Button** object with the PlaceHolder style and size it as wide as a **ComboBox** control. Then place a **ComboBox** control on the placeholder.

Double clicking a toolbar at run time invokes the Customize Toolbar dialog box, which allows the user to hide, display, or rearrange toolbar buttons. To enable or disable the dialog box, use the **AllowCustomize** property. You can also invoke the Customize Toolbar dialog box using the **Customize** method. If you wish to save and restore the state of a toolbar, or allow the user to do so, two methods are provided: the **SaveToolbar** and **RestoreToolbar** methods. The Change event, generated when a toolbar is altered, is typically used to invoke the **SaveToolbar** method.

Usability is further enhanced by programming **ToolTipText** descriptions of each **Button** object. To display ToolTips, the **ShowTips Property** must be set to **True**. When the user invokes the Customize Toolbar dialog box, clicking a button causes a description of the button to be displayed in the dialog box; this description can be programmed by setting the **Description** property.

Distribution Note The **Toolbar** control is part of a group of custom controls that are found in the COMCTL32.OCX file. To use the **Toolbar** control in your application, you must add the COMCTL32.OCX file to the project. When distributing your application, install the COMCTL32.OCX file in the user's Microsoft Windows SYSTEM directory. For more information on how to add a custom control to a project, see the *Programmer's Guide*.

Toolbar Control Example

This example adds **Button** objects to a **Toolbar** control using the **Add** method and assigns images supplied by the **ImageList** control. The behavior of each button is determined by the **Style** property. The code creates buttons that can be used to open and save files and includes a **ComboBox** control that is used to change the backcolor of the form. To try the example, place a **Toolbar**, **ImageList**, and a **ComboBox** on a form and paste the code into the form's Declarations section. Make sure that you insert the **ComboBox** directly on the **Toolbar** control. Run the example, click the various buttons and select from the combo box.

```
Private Sub Form_Load()
    ' Create object variable for the ImageList.
    Dim imgX As ListImage

    ' Load pictures into the ImageList control.
    Set imgX = ImageList1.ListImages. _
    Add(, "open", LoadPicture("bitmaps\tlbr_w95\open.bmp"))
    Set imgX = ImageList1.ListImages. _
    Add(, "save", LoadPicture("bitmaps\tlbr_w95\save.bmp"))
    Toolbar1.ImageList = ImageList1

    ' Create object variable for the Toolbar.
    Dim btnX As Button
    ' Add button objects to Buttons collection using
    ' the
    ' Add method. After creating each button, set both
    ' Description and ToolTipText properties.
    Toolbar1.Buttons.Add , , , tbrSeparator
    Set btnX = Toolbar1.Buttons.Add(, "open", , tbrDefault, "open")
    btnX.ToolTipText = "Open File"
    btnX.Description = btnX.ToolTipText
    Set btnX = Toolbar1.Buttons.Add(, "save", , tbrDefault, "save")
    btnX.ToolTipText = "Save File"
    btnX.Description = btnX.ToolTipText
    Set btnX = Toolbar1.Buttons.Add(, , , tbrSeparator)

    ' The next button has the Placeholder style. A
    ' ComboBox control will be placed on top of this
    ' button.
    Set btnX = Toolbar1.Buttons.Add(, "combo1", , tbrPlaceholder)
    btnX.Width = 1500 ' Placeholder width to accommodate a combobox.

    Show ' Show form to continue configuring ComboBox.

    ' Configure ComboBox control to be at same location
    ' as the
    ' Button object with the PlaceHolder style (key =
    ' "combo1").
    With Combo1
        .Width = Toolbar1.Buttons("combo1").Width
        .Top = Toolbar1.Buttons("combo1").Top
        .Left = Toolbar1.Buttons("combo1").Left
        .AddItem "Black" ' Add colors for text.
        .AddItem "Blue"
        .AddItem "Red"
```

```
        .ListIndex = 0
    End With

End Sub

Private Sub Form_Resize()
    ' Configure ComboBox control.
    With Combo1
        .Width = Toolbar1.Buttons("combo1").Width
        .Top = Toolbar1.Buttons("combo1").Top
        .Left = Toolbar1.Buttons("combo1").Left
    End With

End Sub
Private Sub toolbar1_ButtonClick(ByVal Button As Button)
    ' Use the Key property with the SelectCase statement to specify
    ' an action.
    Select Case Button.Key
    Case Is = "open"           ' Open file.
        MsgBox "Add code to open file here!"
    Case Is = "save"              ' Save file.
        MsgBox "Add code to save file here!"
    End Select
End Sub

Private Sub Combo1_Click()
    ' Change backcolor of form using the ComboBox.
    Select Case Combo1.ListIndex
    Case 0
        Form1.BackColor = vbBlack
    Case 1
        Form1.BackColor = vbBlue
    Case 2
        Form1.BackColor = vbRed
    End Select
End Sub
```

Button Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vastmSelectCase;vbobjButtonC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjButtonX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjButtonP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjButtonM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjButtonE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjButtonS"}

A **Button** object represents an individual button in the **Buttons** collection of a **Toolbar** control.

Remarks

For each **Button** object, you can add text or a bitmap image, or both, from an **ImageList** control, and set properties to change its state and style.

At design time, use the Insert Button and Remove Button buttons on the Buttons tab in the Toolbar Control Properties dialog box to insert and remove **Button** objects from the **Buttons** collection. At run time, you can also add **Button** objects by using the **Add** method of the **Buttons** collection.

At design time and run time, you can set the **Caption**, **Image**, **Value**, **MixedState**, and **ToolTipText** properties to change the appearance of each **Button** object.

Whenever a button is clicked on the **Toolbar** control, the ButtonClick event is called with the selected **Button** object passed in as a parameter. To cause some action to occur when a button is clicked, use the **Index** or **Key** properties in a **Select Case** statement as in the following code:

```
Select Case Button.Key
    Case Is = "open" ' Open file.
    ' Add code to Open a file here
    Case Is = "save" ' Save file.
    ' Add code to Save a file here
    Case Else
    ' If any other button is pressed
End Select
```

Buttons Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolButtonsCollectionC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolButtonsCollectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vamthItem;vbcolButtonsCollectionP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vamthClear;vamthRemove;vbcolButtonsCollectionM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbcolButtonsCollectionE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcolButtonsCollectionS"}

A **Buttons** collection is a collection of **Button** objects for a **Toolbar** control.

Syntax

*toolbar.***Buttons***(index)*
*toolbar.***Buttons.Item**(*index*)

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

Part	Description
toolbar	An object expression that evaluates to a **Toolbar** control.
index	An integer or string that uniquely identifies the object in the collection. The integer is the value of the **Index** property; the string is the value of the **Key** property.

Remarks

The **Buttons** collection is a 1-based collection, which means the collection's **Index** property begins with the number 1 (versus 0 in a 0-based collection).

Each item in the collection can be accessed by its index or unique key. For example, to get a reference to the third **Button** object in a collection, use the following syntax:

```
Dim btnX As Button
   ' Reference by index number.
Set btnX = Toolbar1.Buttons(3)
   ' Or reference by unique key.
Set btnX = Toolbar1.Buttons("third") ' Assuming Key is "third."
   ' Or use Item method.
Set btnX = Toolbar1.Buttons.Item(3)
```

Add Method (Buttons Collection)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthAddButtonsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAddButtonsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAddButtonsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAddButtonsS"}

Adds a **Button** object to a **Buttons** collection and returns a reference to the newly created object. Doesn't support named arguments.

Syntax

object.**Add(***index, key, caption, style, image***)**

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

Part	Description
object	Required. An object expression that evaluates to a **Buttons** collection.
index	Optional. An integer specifying the position where you want to insert the **Button** object. If no *index* is specified, the **Button** is added to the end of the **Buttons** collection.
key	Optional. A unique string that identifies the **Button** object. Use this value to retrieve a specific **Button** object.
caption	Optional. A string that will appear beneath the **Button** object.
style	Optional. The style of the **Button** object. The available styles are detailed in the **Style** Property (**Button** Object).
image	Optional. An integer or unique key that specifies a **ListImage** object in an associated **ImageList** control.

Remarks

You can add **Button** objects at design time using the Buttons tab of the Toolbar Control Properties dialog box. At run time, use the **Add** method to add **Button** objects as in the following code:

```
Dim btnButton as Button
Set btnButton = Toolbar1.Buttons.Add(, "open", , tbrDefault, "open")
```

You associate an **ImageList** control with the **Toolbar** through the **Toolbar** control's **ImageList** property.

AllowCustomize Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAllowCustomizeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAllowCustomizeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAllowCustomizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAllowCustomizeS"}

Returns or sets a value determining if a **Toolbar** control can be customized by the end user with the Customize Toolbar dialog box.

Syntax

object.**AllowCustomize** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to a **Toolbar** control.
boolean	A constant or value that determines if the user can customize a **Toolbar** control, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	Allows the end user to invoke the Customize Toolbar dialog box by double clicking a **Toolbar** control.
False	Customization of the **Toolbar** control with the Customize Toolbar dialog box is not allowed.

Remarks

If the **AllowCustomize** property is set to **True**, double-clicking a **Toolbar** control at run time invokes the Customize Toolbar dialog box.

The Customize Toolbar can also be invoked with the **Customize** method.

ButtonClick Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtButtonClickC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtButtonClickX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtButtonClickA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtButtonClickS"}

Occurs when the user clicks on a **Button** object in a **Toolbar** control.

Syntax

Private Sub *object*_**ButtonClick**(**ByVal** *button* **As Button**)

The ButtonClick event syntax has these parts:

Part	Description
object	An object expression that evaluates to a **Toolbar** control.
button	A reference to the clicked **Button** object.

Remarks

To program an individual **Button** object's response to the ButtonClick event, use the value of the *button* argument. For example, the following code uses the **Key** property of the **Button** object to determine the appropriate action.

```
Private Sub Toolbar1_ButtonClick(ByVal Button As Button)
   Select Case Button.Key
   Case "Open"
      CommonDialog1.ShowOpen
   Case "Save"
      CommonDialog1.ShowSave
   End Select
End Sub
```

Note Because the user can rearrange **Button** objects using the Customize Toolbar dialog box, the value of the **Index** property may not always indicate the position of the button. Therefore, it's preferable to use the value of the **Key** property to retrieve a **Button** object.

ButtonHeight, ButtonWidth Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproButtonHeightC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproButtonHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproButtonHeightA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproButtonHeightS"}

Return or set the height and width of a **Toolbar** control's buttons.

Syntax

object.**ButtonHeight** [= *number*]
object.**ButtonWidth** [= *number*]

The **ButtonHeight, ButtonWidth** properties syntax has these parts:

Part	Description
object	An object expression that evaluates to a **Toolbar** control.
number	A numeric expression specifying the dimensions of all buttons on the control that have the Button, Check, or ButtonGroup style.

Remarks

ButtonHeight and **ButtonWidth** use the scale unit of the Toolbar control's container. The scale unit is determined by the **ScaleMode** property of the container**.**

By default, the **ButtonWidth** and **ButtonHeight** properties are automatically updated to accommodate the string in the **Caption** property or image in the **Image** property of the **Button** object.

Buttons Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproButtonsC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproButtonsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproButtonsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproButtonsS"}

Returns a reference to a **Toolbar** control's collection of **Button** objects.

Syntax

object.**Buttons**

The *object* placeholder is an object expression that evaluates to a **Toolbar** control.

Remarks

You can manipulate **Button** objects using standard collection methods (for example, the **Add** and **Remove** methods). Each element in the collection can be accessed by its index, the value of the **Index** property, or by a unique key, the value of the **Key** property.

Customize Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthCustomizeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthCustomizeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthCustomizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthCustomizeS"}

Invokes the Customize Toolbar dialog box which allows the end user to rearrange or hide **Button** objects on a **Toolbar** control.

Syntax

object.**Customize**

The *object* placeholder is an object expression that evaluates to a **Toolbar** control.

Remarks

The **Toolbar** control contains a built-in dialog box that allows the user to hide, display, or rearrange buttons on a toolbar. Double-clicking the toolbar calls the **Customize** method, which invokes the dialog box.

Use the **Customize** method when you wish to restrict the alteration of the toolbar. For example, the code below allows the user to customize the toolbar only if a password is given:

```
Private Sub Command1_Click()
   If InputBox("Password:") = "Chorus&Line9" Then
      Toolbar1.Customize   ' Invoke Customize method.
   End If
End Sub
```

To preserve the state of a **Toolbar** control, the **SaveToolbar** method writes to the Windows registry. You can restore a **Toolbar** control to a previous state using the **RestoreToolbar** method to read the information previously saved in the registry.

Description Property (Button Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDescriptionToolbarButtonC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDescriptionToolbarButtonX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDescriptionToolbarButtonA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDescriptionToolbarButtonS"}

Returns or sets the text for a **Button** object's description, which is displayed in the Customize Toolbar dialog box.

Syntax

object.**Description** [= *string*]

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

Part	Description
object	An object expression that evaluates to a **Button** object.
string	The string displayed in the Customize Toolbar dialog box when the button is selected.

Remarks

At run time, the Customize Toolbar dialog box can be invoked either by a user double-clicking the **Toolbar** control or programmatically using the **Customize** method. In either case, when the user selects a button in the dialog box, a description of the button is displayed in the lower-left corner of the dialog box. The text for that description is set with the **Description** property.

You can set the **Description** text when you add a **Button** object, as follows:

```
Dim btnX As Button
' Set Image property to a button with the Key "save."
Set btnX = Toolbar1.Buttons.Add(,"save")
btnX.Description = "Save a file."
```

MixedState Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMixedStateC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMixedStateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMixedStateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMixedStateS"}

Returns or sets a value that determines if a **Button** object in a **Toolbar** control appears in an indeterminate state.

Syntax

object.**MixedState** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to a **Button** object.
boolean	A Boolean expression that determines if a **Button** shows the indeterminate state, as specified in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The **Button** object is in the indeterminate state and becomes dimmed.
False	The **Button** object is not in the indeterminate state and looks normal.

Remarks

The **MixedState** property is typically used when a selection contains a variety of attributes. For example, if you select text that contains both plain (normal) characters and bold characters, the **MixedState** property is used. The image displayed by the **Button** object could then be changed to indicate its state, which would differ from the Checked and Unchecked value returned by the **Value** property.

RestoreToolbar Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRestoreToolbarC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRestoreToolbarX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRestoreToolbarA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRestoreToolbarS"}

Restores a toolbar, created with a **Toolbar** control, to its original state after being customized. Doesn't support named arguments.

Syntax

object.**RestoreToolbar(***key* **As String***, subkey* **As String**, *value* **As String)**

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

Part	Description
object	Required. An object expression that evaluates to a **Toolbar** control.
key	Required. A string expression that specifies the key in the Windows registry where the method retrieves the **Toolbar** information.
subkey	Required. A string expression that specifies a subkey under the *key* parameter in the registry.
value	Required. A string expression that identifies the value under the *subkey* where the **Toolbar** information is stored in the registry.

Remarks

To customize the **Toolbar** control at run time, use the **Customize** method in code or if the **AllowCustomize** property is **True**, the user can customize it by double clicking the control.

The state of the toolbar can be saved in the registry using the **SaveToolbar** method. The **RestoreToolbar** method restores the state of a toolbar by reading the registry.

The following code restores the **Toolbar** control's settings for the current user, assuming they have previously been saved with the **SaveToolbar** method.

```
Toolbar1.RestoreToolbar "AppName", "User1", "Toolbar1"
```

SaveToolbar Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSaveToolbarC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSaveToolbarX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSaveToolbarA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSaveToolbarS"}

At run time, saves the state of a toolbar, created with the **Toolbar** control, in the registry. Doesn't support named arguments.

Syntax

object.**SaveToolbar(***key* **As String,** *subkey* **As String**, *value* **As String)**

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

Part	Description
object	Required. An object expression that evaluates to a **Toolbar** control.
key	Required. A string expression specifying the key in the registry where the method stores the **Toolbar** information.
subkey	Required. A string expression that specifies a location in the registry under the *key* parameter.
value	Required. The **Toolbar** information to be stored in the *subkey*.

Remarks

To customize the **Toolbar** control at run time, use the **Customize** method in code or if the **AllowCustomize** property is **True**, the user can customize it by double clicking the control.

If the *key*, *subkey, or value* you specify doesn't exist in the registry, it is created.

To save more than one version of the toolbar, you can change the *subkey* or *value* parameter. This causes the toolbar to write to a different part of the registry. The following code saves two different states of a toolbar after it has been customized.

```
' Save settings for User1
Toolbar1.SaveToolbar "AppName", "User1", "Toolbar1"

' Save settings for User2
Toolbar1.SaveToolbar "AppName", "User2", "Toolbar1"
```

Since the Change event for the **Toolbar** control occurs after the toolbar has been customized, in most cases the above code can be placed in the Change event for the toolbar.

Style Property (Button Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproStyleButtonC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproStyleButtonX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStyleButtonA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStyleButtonS"}

Returns or sets a constant or value that determines the appearance and behavior of a **Button** object in a **Toolbar** control.

Syntax

object.**Style** [=*value*]

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

Part	Description
object	An object expression that evaluates to a **Button** object.
value	A constant or integer that determines the appearance and behavior of a **Button** object, as specified in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
tbrDefault	0	(Default) Button. The button is a regular push button.
tbrCheck	1	Check. The button is a check button, which can be checked or unchecked.
tbrButtonGroup	2	ButtonGroup. The button remains pressed until another button in the group is pressed. Exactly one button in the group can be pressed at any one moment.
tbrSeparator	3	Separator. The button functions as a separator with a fixed width of 8 pixels.
tbrPlaceholder	4	Placeholder. The button is like a separator in appearance and functionality, but has a settable width.

Remarks

Buttons that have the ButtonGroup style must be grouped. To distinguish a group, place all **Button** objects with the same style (ButtonGroup) between two **Button** objects with the Separator style.

You can also place another control on a toolbar by assigning a **Button** object the PlaceHolder style, then drawing a control on to the toolbar. For example, to place a drop-down combo box on a toolbar at design time, add a **Button** object with the PlaceHolder style and size it to the size of a **ComboBox** control. Then place a **ComboBox** on the placeholder.

When a **Button** object is assigned the PlaceHolder style, you can set the value of the **Width** property to accommodate another control placed on the **Button**. If a **Button** object has the Button, Check, or ButtonGroup style, the height and width are determined by the **ButtonHeight** and **ButtonWidth** properties.

If you place a control on a button with the PlaceHolder style, you must use code to align and size the control if the form is resized, as shown below:

```
Private Sub Form_Resize()
   ' Track a ComboBox by setting its Top, Left, and
   ' Width properties
   ' to the Top, Left, and Width properties of a
   ' Button object
With Toolbar1.Buttons("Combo1")
   Combo1.Move .Left,.Top,.Width
End With
End Sub
```

Wrappable Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproWrappableC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWrappableX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWrappableA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWrappableS"}

Returns or sets a value that determines if **Toolbar** control buttons will automatically wrap when the window is resized.

Syntax

object.**Wrappable** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to a **Toolbar** control.
boolean	A Boolean expression that determines if the **Button** objects on a **Toolbar** control will wrap, as described in Settings.

Settings

The settings for *boolean* are:

Value	Description
True	The buttons on the **Toolbar** control wrap if the form is resized.
False	The buttons on the **Toolbar** control won't wrap if the form is resized.

Toolbar Control Constants

Style Constants

Constant	Value	Description
tbrDefault	0	The button is a regular push button.
tbrCheck	1	The button is a check button.
tbrButtonGroup	2	The button remains pressed until another button in the group is pressed. Exactly one button in the group is pressed at any time.
tbrSeparator	3	The button functions as a separator with a fixed width of 8 pixels.
tbrPlaceholder	4	The button is like a separator in appearance and functionality but has a settable width.

Value Constants

Constant	Value	Description
tbrUnpressed	0	The button is not currently pressed or checked.
tbrPressed	1	The button is currently pressed or checked.

Controls Property (Toolbar Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthControlsCollectionPropertyC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthControlsCollectionPropertyX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthControlsCollectionPropertyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproControlsCollectionPropertyS"}

Returns a reference to a collection of controls contained on an object.

Syntax

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

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	A value that identifies a member of a Controls collection.

Remarks

The **Controls** property is similar to the **Controls** collection on the **Form** object, and is accessed in a similar manner. For example, use the following code to get the **Top** property of the second control on a **Toolbar** control:

```
MsgBox Toolbar1.Controls(2).Top
```

With the **Controls** property, you can iterate through all the controls on a **Tab** object or a **Toolbar** control and change the properties of each control as in the following code:

```
For Each ContainedControl in Toolbar1.Controls
   ContainedControl.Width = Toolbar1.Width / Toolbar1.Buttons.Count
Next
```

Note The **Controls** collection refers to controls contained by the **Toolbar** control, such as a **ComboBox** control, and not the **Button** objects, which are part of the control itself.

TreeView Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjTreeViewC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjTreeViewX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjTreeviewP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjTreeviewVbmth"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjTreeviewE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjTreeViewS"}

A **TreeView** control displays a hierarchical list of **Node** objects, each of which consists of a label and an optional bitmap. A **TreeView** is typically used to display the headings in a document, the entries in an index, the files and directories on a disk, or any other kind of information that might usefully be displayed as a hierarchy.

Syntax

Treeview

Remarks

After creating a **TreeView** control, you can add, remove, arrange, and otherwise manipulate **Node** objects by setting properties and invoking methods. You can programmatically expand and collapse **Node** objects to display or hide all child nodes. Three events, the Collapse, Expand, and NodeClick event, also provide programming functionality.

You can navigate through a tree in code by retrieving a reference to **Node** objects using **Root**, **Parent**, **Child**, **FirstSibling**, **Next**, **Previous**, and **LastSibling** properties. Users can navigate through a tree using the keyboard as well. UP ARROW and DOWN ARROW keys cycle downward through all expanded **Node** objects. **Node** objects are selected from left to right, and top to bottom. At the bottom of a tree, the selection jumps back to the top of the tree, scrolling the window if necessary. RIGHT ARROW and LEFT ARROW keys also tab through expanded **Node** objects, but if the RIGHT ARROW key is pressed while an unexpanded **Node** is selected, the **Node** expands; a second press will move the selection to the next **Node**. Conversely, pressing the LEFT ARROW key while an expanded **Node** has the focus collapses the **Node**. If a user presses an ANSI key, the focus will jump to the nearest **Node** that begins with that letter. Subsequent pressings of the key will cause the selection to cycle downward through all expanded nodes that begin with that letter.

Several styles are available which alter the appearance of the control. **Node** objects can appear in one of eight combinations of text, bitmaps, lines, and plus/minus signs.

The **TreeView** control uses the **ImageList** control, specified by the **ImageList** property, to store the bitmaps and icons that are displayed in **Node** objects. A **TreeView** control can use only one **ImageList** at a time. This means that every item in the **TreeView** control will have an equal-sized image next to it when the **TreeView** control's **Style** property is set to a style which displays images.

Distribution Note The **TreeView** control is part of a group of ActiveX controls that are found in the COMCTL32.OCX file. To use the **TreeView** control in your application, you must add the COMCTL32.OCX file to the project. When distributing your application, install the COMCTL32.OCX file in the user's Microsoft Windows System or System32 directory.

TreeView Control Constants

TreeLine Constants

Constant	Value	Description
tvwTreeLines	0	Treelines shown.
tvwRootLines	1	Rootlines shown with Treelines.

TreeRelationship Constants

Constant	Value	Description
tvwFirst	0	First Sibling.
tvwLast	1	Last Sibling.
tvwNext	2	Next sibling.
tvwPrevious	3	Previous sibling.
tvwChild	4	Child.

TreeStyle Constants

Constant	Value	Description
tvwTextOnly	0	Text only.
tvwPictureText	1	Picture and text.
tvwPlusMinusText	2	Plus/minus and text.
tvwPlusPictureText	3	Plus/minus, picture, and text.
tvwTreelinesText	4	Treelines and text.
tvwTreelinesPictureText	5	Treelines, Picture, and Text.
tvwTreelinesPlusMinusText	6	Treelines, Plus/Minus, and Text.
tvwTreelinesPlusMinusPictureText	7	Treelines, Plus/Minus, Picture, and Text.

LabelEdit Constants

Constant	Value	Description
tvwAutomatic	0	Label Editing is automatic.
tvwManual	1	Label Editing must be invoked.

Node Object, Nodes Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjNodeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjNodeX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vamthItem;vbobjNodeP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vamthClear;vamthRemove;vbobjNodeM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjNodeE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjNodeS"}

- A **Node** object is an item in a **TreeView** control that can contain images and text.
- A **Nodes** collection contains one or more **Node** objects.

Syntax

treeview.**Nodes**
treeview.**Nodes.Item(***index***)**

The syntax lines above refer to the collection and to individual elements in the collection, respectively, according to standard collection syntax.

The **Node** object and **Nodes** collection syntax have these parts:

Part	Description
treeview	An object expression that evaluates to a TreeView control.
index	Either an integer or string that uniquely identifies a member of a **Nodes** collection. The integer is the value of the **Index** property; the string is the value of the **Key** property.

Remarks

Nodes can contain both text and pictures. However, to use pictures, you must associate an **ImageList** control using the **ImageList** property.

Pictures can change depending on the state of the node; for example, a selected node can have a different picture from an unselected node if you set the **SelectedImage** property to an image from the associated **ImageList**.

Add Method (Nodes Collection)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vamthItem;vbmthAddNodeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAddNodeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAddNodeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAddNodeS"}

Adds a **Node** object to a **Treeview** control's **Nodes** collection. Doesn't support named arguments.

Syntax

object.**Add(***relative, relationship, key, text, image, selectedimage***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
relative	Optional. The index number or key of a pre-existing **Node** object. The relationship between the new node and this pre-existing node is found in the next argument, *relationship*.
relationship	Optional. Specifies the relative placement of the **Node** object, as described in Settings.
key	Optional. A unique string that can be used to retrieve the **Node** with the **Item** method.
text	Required. The string that appears in the **Node**.
image	Optional. The index of an image in an associated **ImageList** control.
selectedimage	Optional. The index of an image in an associated **ImageList** control that is shown when the **Node** is selected.

Settings

The settings for *relationship* are:

Constant	Value	Description
tvwFirst	0	First. The **Node** is placed before all other nodes at the same level of the node named in *relative*.
tvwLast	1	Last. The **Node** is placed after all other nodes at the same level of the node named in *relative*. Any Node added subsequently may be placed after one added as Last.
tvwNext	2	Next. The **Node** is placed after the node named in *relative*.
tvwPrevious	3	Previous. The **Node** is placed before the node named in *relative*.
tvwChild	4	(Default) Child. The **Node** becomes a child node of the node named in *relative*.

Note If no **Node** object is named in *relative*, the new node is placed in the last position of the top node hierarchy.

Remarks

The **Nodes** collection is a 1-based collection.

As a **Node** object is added it is assigned an index number, which is stored in the **Node** object's **Index** property. This value of the newest member is the value of the **Node** collection's **Count** property.

Because the **Add** method returns a reference to the newly created **Node** object, it is most convenient to set properties of the new **Node** using this reference. The following example adds several **Node** objects with identical properties:

```
Dim nodX As Node  ' Declare the object variable.
Dim I as Integer  ' Declare a counter variable.
For I = 1 to 4
   Set nodX = TreeView1.Nodes.Add(,,,"Node " & Cstr(i))
   ' Use the reference to set other properties, such as Enabled.
   nodX.Enabled = True
   ' Set image property to image 3 in an associated ImageList.
   nodX.ExpandedImage = 3
Next I
```

Add Method Example (Nodes Collection)

The following example adds two **Node** objects to a **TreeView** control. To try the example, place a **TreeView** control on a form, and paste the code into the form's Declarations section. Run the example, and click the **Node** object to expand it.

```
Private Sub Form_Load()
   ' Set Treeview control properties.
   TreeView1.LineStyle = tvwRootLines  ' Linestyle 1

   ' Add Node objects.
   Dim nodX As Node    ' Declare Node variable.
   ' First node with 'Root' as text.
   Set nodX = TreeView1.Nodes.Add(, , "r", "Root")

   ' This next node is a child of Node 1 ("Root").
   Set nodX = TreeView1.Nodes.Add("r", tvwChild, "child1", "Child")

End Sub
```

Child Property (TreeView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproChildNodeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproChildNodeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproChildNodeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproChildNodeS"}

Returns a reference to the first child of a **Node** object in a **TreeView** control.

Syntax

object.**Child**

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

Remarks

The **Child**, **FirstSibling**, **LastSibling**, **Previous**, **Parent**, **Next**, and **Root** properties all return a reference to another **Node** object. Therefore, you can simultaneously reference and perform operations on a **Node**, as follows:

```
With TreeView1.Nodes(TreeView1.SelectedItem.Index).Child
   .Text = "New text"
   .Key = "New key"
   .SelectedImage = 3
End With
```

You can also set an object variable to the referenced **Node**, as follows:

```
Dim NodChild As Node
' Get a reference to the child of the selected node.
Set NodChild = TreeView1.Nodes(TreeView1.SelectedItem.Index).Child
' Use this reference to perform operations on the child Node.
With nodChild
   .Text = "New text"    '  Change the text.
   .Key = "New key"  ' Change key.
   .SelectedImage = 3    ' Change SelectedImage.
End With
```

Child Property Example

This example creates several **Node** objects. When you click on a **Node** object, the code first uses the **Children** property to determine if the **Node** has children nodes. If so, the caption of the form displays the text of the **Child** node.

```
Option Explicit

Private Sub Form_Load()
' This code creates a tree with 3 Node objects.
   TreeView1.Style = tvwTreelinesPlusMinusText ' Style 6.
   TreeView1.LineStyle = tvwRootLines  'Linestyle 1.

   ' Add several Node objects.
   Dim nodX As Node    ' Create variable.

   Set nodX = TreeView1.Nodes.Add(, , "r", "Root")
   Set nodX = TreeView1.Nodes.Add("r", tvwChild, "c1", "Child 1")

   nodX.EnsureVisible ' Show all nodes.
   Set nodX = TreeView1.Nodes.Add("c1", tvwChild, "c2", "Child 2")
   Set nodX = TreeView1.Nodes.Add("c1", tvwChild, "c3", "Child 3")
   nodX.EnsureVisible ' Show all nodes.
End Sub

Private Sub TreeView1_NodeClick(ByVal Node As Node)
   ' If the Node does have children, then display the text of
   ' the child Node.
   If Node.Children Then
      Caption = Node.Child.Text
   End If
End Sub
```

Children Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproChildrenC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproChildrenX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproChildrenA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproChildrenS"}

Returns the number of child **Node** objects contained in a **Node** object.

Syntax

object.**Children**

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

Remarks

The **Children** property can be used to check if a **Node** object has any children before performing an operation that affects the children. For example, the following code checks for the presence of child nodes before retrieving the **Text** property of the first **Node**, using the **Child** property.

```
Private Sub TreeView1_NodeClick(ByVal Node As Node)
   If Node.Children > 0 Then
      MsgBox Node.Child.Text
   End If
End Sub
```

Children Property Example

This example puts several **Node** objects in a **TreeView** control. The code checks to see if a **Node** has children nodes. If so, then it displays the text of the children nodes. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, click a **Node** object to select it, then click the form to see the text of the **Node** object's children.

```
Option Explicit
Private Sub Form_Click()
   Dim strC As String
   Dim N As Integer
   If TreeView1.SelectedItem.Children > 0 Then ' There are children.

      ' Get first child's text, and set N to its index value.
      strC = TreeView1.SelectedItem.Child.Text & vbLF
      N = TreeView1.SelectedItem.Child.Index

      ' While N is not the index of the child node's
      ' last sibling, get next sibling's text.
      While N <> TreeView1.SelectedItem.Child.LastSibling.Index
         strC = strC & TreeView1.Nodes(N).Next.Text & vbLF
         ' Reset N to next sibling's index.
         N = TreeView1.Nodes(N).Next.Index
      Wend
      ' Show results.
      MsgBox "Children of " & TreeView1.SelectedItem.Text & _
      " are: " & vbLF & strC
   Else ' There are no children.
      MsgBox TreeView1.SelectedItem.Text & " has no children"
   End If
End Sub

Private Sub Form_Load()
   TreeView1.BorderStyle = 1  ' Ensure border is visible
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,"d","Dates")
   Set nodX = TreeView1.Nodes.Add("d",tvwChild,"d89","1989")
   Set nodX = TreeView1.Nodes.Add("d",tvwChild,"d90","1990")

   ' Create children of 1989 node.
   Set nodX = TreeView1.Nodes.Add("d89",tvwChild, ,"John")
   Set nodX = TreeView1.Nodes.Add("d89",tvwChild, ,"Brent")
   Set nodX = TreeView1.Nodes.Add("d89",tvwChild, ,"Eric")
   Set nodX = TreeView1.Nodes.Add("d89",tvwChild, ,"Ian")
   nodX.EnsureVisible ' Show all nodes.

   ' Create children of 1990 node.
   Set nodX = TreeView1.Nodes.Add("d90",tvwChild, ,"Randy")
   Set nodX = TreeView1.Nodes.Add("d90",tvwChild, ,"Ron")
   nodX.EnsureVisible ' Show all nodes.
End Sub
```

Collapse Event (TreeView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtCollapseC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtCollapseX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtCollapseA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtCollapseS"}

Generated when any **Node** object in a **TreeView** control is collapsed.

Syntax

Private Sub *object*_**Collapse(ByVal** *node* **As Node)**

The Collapse event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
node	A reference to the clicked **Node** object.

Remarks

The Collapse event occurs before the standard Click event.

There are three methods of collapsing a **Node**: by setting the **Node** object's **Expanded** property to **False**, by double-clicking a **Node** object, and by clicking a plus/minus image when the **TreeView** control's **Style** property is set to a style that includes plus/minus images. All of these methods generate the Collapse event.

The event passes a reference to the collapsed **Node** object. The reference can validate an action, as in the following example:

```
Private Sub TreeView1_Collapse(ByVal Node As Node)
   If Node.Index = 1 Then
      Node.Expanded = True ' Expand the node again.
   End If
End Sub
```

Collapse Event (TreeView Control) Example

This example adds one **Node** object, with several child nodes, to a **TreeView** control. When the user collapses a **Node,** the code checks to see how many children the **Node** has. If it has more than one child, the **Node** is re-expanded. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and double-click a **Node** to collapse it and generate the event.

```
Private Sub Form_Load()
   TreeView1.Style = tvwTreelinesPlusMinusText ' Style 6.
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,"DV","Da Vinci")
   Set nodX = TreeView1.Nodes.Add("DV",tvwChild,"T","Titian")
   Set nodX = TreeView1.Nodes.Add("T",tvwChild,"R","Rembrandt")
   Set nodX = TreeView1.Nodes.Add("R",tvwChild,,"Goya")
   Set nodX = TreeView1.Nodes.Add("R",tvwChild,,"David")
   nodX.EnsureVisible   ' Show all nodes.
End Sub

Private Sub TreeView1_Collapse(ByVal Node As Node)
   ' If the Node has more than one child node,
   ' keep the node expanded.
   Select Case Node.Children
      Case Is > 1
         Node.Expanded = True
   End Select
End Sub
```

AfterLabelEdit Event (ListView, TreeView Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtAfterLabelEditC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"svbevtAfterLabelEditX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtAfterLabelEditA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtAfterLabelEditS"}

Occurs after a user edits the label of the currently selected **Node** or **ListItem** object.

Syntax

Private Sub *object*_**AfterLabelEdit(***cancel* **As Integer**, *newstring* **As String)**

The AfterLabelEdit event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
cancel	An integer that determines if the label editing operation is canceled. Any nonzero integer cancels the operation. Boolean values are also accepted.
newstring	The string the user entered, or **Null** if the user canceled the operation.

Remarks

Both the AfterLabelEdit and the BeforeLabelEdit events are generated only if the **LabelEdit** property is set to 0 (Automatic), or if the **StartLabelEdit** method is invoked.

The AfterLabelEdit event is generated after the user finishes the editing operation, which occurs when the user clicks on another **Node** or **ListItem** or presses the ENTER key.

To cancel a label editing operation, set *cancel* to any nonzero number or to **True**. If a label editing operation is canceled, the previously existing label is restored.

The *newstring* argument can be used to test for a condition before canceling an operation. For example, the following code cancels the operation if *newstring* is a number:

```
Private Sub TreeView1_AfterLabelEdit(Cancel As Integer, NewString As
String)
   If IsNumeric(NewString) Then
      MsgBox "No numbers allowed"
      Cancel = True
   End If
End Sub
```

AfterLabelEdit Event (ListView, TreeView Controls) Example

This example adds three **Node** objects to a **TreeView** control. When you attempt to edit a **Node** object's label, the object's index is checked. If it is 1, the operation is canceled. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, click twice on the top **Node** object's label to edit it, type in some text, and press ENTER.

```
Private Sub Form_Load()
   TreeView1.Style = tvwTreelinesText ' Lines and text.
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,,"Parent")
   Set nodX = TreeView1.Nodes.Add(1,tvwChild,,"Child1")
   Set nodX = TreeView1.Nodes.Add(1,tvwChild,,"Child2")
   nodX.EnsureVisible   ' Make sure all nodes are visible.
End Sub

Private Sub TreeView1_AfterLabelEdit _
(Cancel As Integer, NewString As String)
   ' If current node's index is 1, edit is canceled.
   If TreeView1.SelectedItem.Index = 1 Then
      Cancel = True
      MsgBox "Can't replace " & TreeView1.SelectedItem.Text & _
      " with " & NewString
   End If
End Sub
```

This example adds three **ListItem** objects to a **ListView** control. When you attempt to edit a **ListItem** object's label, the object's index is checked. If it is 1, the operation is canceled. To try the example, place a **ListView** control on a form and paste the code into the form's Declarations section. Run the example, click twice on any **ListItem** object's label to edit it, type in some text, and press ENTER.

```
Private Sub Form_Load()
   Dim itmX As ListItem
   Set itmX = ListView1.ListItems.Add(,,"Item1")
   Set itmX = ListView1.ListItems.Add(,,"Item 2")
   Set itmX = ListView1.ListItems.Add(,,"Item 3")
End Sub

Private Sub ListView1_AfterLabelEdit _
(Cancel As Integer, NewString As String)
   ' If current ListItem's index is 1, edit is canceled.
   If ListView1.SelectedItem.Index = 1 Then
      Cancel = True
      MsgBox "Can't replace " & ListView1.SelectedItem.Text & _
      " with " & NewString
   End If
End Sub
```

BeforeLabelEdit Event (ListView, TreeView Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtBeforeLabelEditC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtBeforeLabelEditX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtBeforeLabelEditA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtBeforeLabelEditS"}

Occurs when a user attempts to edit the label of the currently selected **ListItem** or **Node** object.

Syntax

Private Sub *object*_**BeforeLabelEdit(***cancel* **As Integer)**

The BeforeLabelEdit event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
cancel	An integer that determines if the operation is canceled. Any nonzero integer cancels the operation. The default is 0.

Remarks

Both the AfterLabelEdit and the BeforeLabelEdit events are generated only if the **LabelEdit** property is set to 0 (Automatic), or if the **StartLabelEdit** method is invoked.

The BeforeLabelEdit event occurs after the standard **Click** event.

To begin editing a label, the user must first click the object to select it, and click it a second time to begin the operation. The BeforeLabelEdit event occurs after the second click.

To determine which object's label is being edited, use the **SelectedItem** property. The following example checks the index of a selected **Node** before allowing an edit. If the index is 1, the operation is cancelled.

```
Private Sub TreeView1_BeforeLabelEdit(Cancel As Integer)
   If TreeView1.SelectedItem.Index = 1 Then
      Cancel = True  ' Cancel the operation
   End If
End Sub
```

BeforeLabelEdit Event (ListView, TreeView Controls) Example

This example adds several **Node** objects to a **TreeView** control. If you try to edit a label, the **Node** object's index is checked. If it is 1, the edit is prevented. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and try to edit the labels.

```
Private Sub Form_Load()
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,"P1","Parent 1")
   Set nodX = TreeView1.Nodes.Add("P1",tvwChild,,"Child 1")
   Set nodX = TreeView1.Nodes.Add("P1",tvwChild,,"Child 2")
   nodX.EnsureVisible   ' Make sure all nodes are visible.
End Sub

Private Sub TreeView1_BeforeLabelEdit(Cancel As Integer)
   ' Check selected node's index. If it is 1,
   ' then cancel the editing operation.
   If TreeView1.SelectedItem.Index = 1 Then
      MsgBox "Can't edit " + TreeView1.SelectedItem.Text
      Cancel = True
   End If
End Sub
```

This example adds several **ListItem** objects to a **ListView** control. If you try to edit a label, the **ListItem** object's index is checked. If it is 1, the edit is prevented. To try the example, place a **ListView** control on a form and paste the code into the form's Declarations section. Run the example, and try to edit the labels.

```
Private Sub Form_Load()
   Dim nodX As ListViewItem
   Set nodX = ListView1.ListItems.Add(, , "Item 1")
   Set nodX = ListView1.ListItems.Add(, , "Item 2")
   Set nodX = ListView1.ListItems.Add(, , "Item 3")
End Sub

Private Sub ListView1_BeforeLabelEdit(Cancel As Integer)
   ' Check selected item's index. If it is 1,
   ' then cancel the editing operation.
   If ListView1.SelectedItem.Index = 1 Then
      MsgBox "Can't edit " + ListView1.SelectedItem.Text
         Cancel = True
   End If
End Sub
```

CreateDragImage Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthCreateDragImageC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthCreateDragImageX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthCreateDragImageA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthCreateDragImageS"}

Creates a drag image using a dithered version of an object's associated image. This image is typically used in drag-and-drop operations.

Syntax

object.**CreateDragImage**

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

Remarks

The **CreateDragImage** method is typically used to assign an image to a **DragIcon** property at the start of a drag-and-drop operation.

CreateDragImage Method Example

This example adds several **Node** objects to a **TreeView** control. After you select a **Node** object, you can drag it to any other **Node**. To try the example, place **TreeView** and **ImageList** controls on a form and paste the code into the form's Declaration section. Run the example and drag **Node** objects around to see the result.

```
' Declare global variables.
Dim indrag As Boolean ' Flag that signals a Drag Drop operation.
Dim nodX As Object ' Item that is being dragged.

Private Sub Form_Load()
   ' Load a bitmap into an Imagelist control.
   Dim imgX As ListImage
   Dim BitmapPath As String
   BitmapPath = "icons\mail\mail01a.ico"
   Set imgX = imagelist1.ListImages.Add(, , LoadPicture(BitmapPath))

   ' Initialize TreeView control and create several nodes.
   TreeView1.ImageList = imagelist1
   Dim nodX As Node  ' Create a tree.
   Set nodX = TreeView1.Nodes.Add(, , , "Parent1", 1)
   Set nodX = TreeView1.Nodes.Add(, , , "Parent2", 1)
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Child 1", 1)
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Child 2", 1)
   Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 3", 1)
   Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 4", 1)
   Set nodX = TreeView1.Nodes.Add(3, tvwChild, , "Child 5", 1)
   nodX.EnsureVisible ' Expand tree to show all nodes.
End Sub

Private Sub TreeView1_MouseDown_
(Button As Integer, Shift As Integer, x As Single, y As Single)
   Set nodX = TreeView1.SelectedItem ' Set the item being dragged.
End Sub

Private Sub TreeView1_MouseMove _
(Button As Integer, Shift As Integer, x As Single, y As Single)
   If Button = vbLeftButton Then ' Signal a Drag operation.
      indrag = True ' Set the flag to true.
      ' Set the drag icon with the CreateDragImage method.
      TreeView1.DragIcon = TreeView1.SelectedItem.CreateDragImage
      TreeView1.Drag vbBeginDrag ' Drag operation.
   End If
End Sub

Private Sub TreeView1_DragDrop_
(Source As Control, x As Single, y As Single)
   If TreeView1.DropHighlight Is Nothing Then
      Set TreeView1.DropHighlight = Nothing
      indrag = False
      Exit Sub
   Else
      If nodX = TreeView1.DropHighlight Then Exit Sub
      Cls
      Print nodX.Text & " dropped on " & TreeView1.DropHighlight.Text
      Set TreeView1.DropHighlight = Nothing
```

```
      indrag = False
   End If
End Sub

Private Sub TreeView1_DragOver(Source As Control, x As Single, y As Single,
State As Integer)
   If indrag = True Then
      ' Set DropHighlight to the mouse's coordinates.
      Set TreeView1.DropHighlight = TreeView1.HitTest(x, y)
   End If
End Sub
```

DropHighlight Property (ListView, TreeView Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDropHighlightC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDropHighlightX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDropHighlightA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDropHighlightS"}

Returns or sets a reference to a **Node** or **ListItem** object that is highlighted with the system highlight color when the cursor moves over it.

Syntax

object.**DropHighlight** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **Node** or **ListItem** object.

Remarks

The **DropHighlight** property is typically used in combination with the **HitTest** method in drag-and-drop operations. As the cursor is dragged over a **ListItem** or **Node** object, the **HitTest** method returns a reference to any object it is dragged over. In turn, the **DropHighlight** property is set to the hit object, and simultaneously returns a reference to that object. The **DropHighlight** property then highlights the hit object with the system highlight color. The following code sets the **DropHighlight** property to the object hit with the **HitTest** method.

```
Private Sub TreeView1_DragOver _
(Source As Control, X As Single, Y As Single, State As Integer)
   Set TreeView1.DropHighlight = TreeView1.HitTest(X,Y)
End Sub
```

Subsequently, you can use the **DropHighlight** property in the DragDrop event to return a reference to the last object the source control was dropped over, as shown in the following code:

```
Private Sub TreeView1_DragDrop _
(Source As Control, x As Single, y As Single)
   ' DropHighlight returns a reference to object drop occurred over.
   Me.Caption = TreeView1.DropHighlight.Text
   ' To release the DropHighlight reference, set it to Nothing.
   Set TreeView1.DropHighlight = Nothing
End Sub
```

Note that in the preceding example, the **DropHighlight** property is set to Nothing after the procedure is completed. This must be done to release the highlight effect.

DropHighlight Property Example

This example adds several **Node** objects to a **TreeView** control. After you select a **Node** object, you can drag it to any other **Node**. To try the example, place **TreeView** and **ImageList** controls on a form and paste the code into the form's Declaration section. Run the example and drag **Node** objects around to see the result.

```
' Declare global variables.
Dim indrag As Boolean ' Flag that signals a Drag Drop operation.
Dim nodX As Object ' Item that is being dragged.

Private Sub Form_Load()
   ' Load a bitmap into an Imagelist control.
   Dim imgX As ListImage
   Dim BitmapPath As String
   BitmapPath = "icons\mail\mail01a.ico"
   Set imgX = imagelist1.ListImages.Add(, , LoadPicture(BitmapPath))

   ' Initialize TreeView control and create several nodes.
   TreeView1.ImageList = imagelist1
   Dim nodX As Node  ' Create a tree.
   Set nodX = TreeView1.Nodes.Add(, , , "Parent1", 1)
   Set nodX = TreeView1.Nodes.Add(, , , "Parent2", 1)
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Child 1", 1)
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Child 2", 1)
   Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 3", 1)
   Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 4", 1)
   Set nodX = TreeView1.Nodes.Add(3, tvwChild, , "Child 5", 1)
   nodX.EnsureVisible ' Expand tree to show all nodes.
End Sub

Private Sub TreeView1_MouseDown_
(Button As Integer, Shift As Integer, x As Single, y As Single)
   Set nodX = TreeView1.SelectedItem ' Set the item being dragged.
End Sub

Private Sub TreeView1_MouseMove _
(Button As Integer, Shift As Integer, x As Single, y As Single)
   If Button = vbLeftButton Then ' Signal a Drag operation.
      indrag = True ' Set the flag to true.
      ' Set the drag icon with the CreateDragImage method.
      TreeView1.DragIcon = TreeView1.SelectedItem.CreateDragImage
      TreeView1.Drag vbBeginDrag ' Drag operation.
   End If
End Sub

Private Sub TreeView1_DragDrop_
(Source As Control, x As Single, y As Single)
   If TreeView1.DropHighlight Is Nothing Then
      Set TreeView1.DropHighlight = Nothing
      indrag = False
      Exit Sub
   Else
      If nodX = TreeView1.DropHighlight Then Exit Sub
      Cls
      Print nodX.Text & " dropped on " & TreeView1.DropHighlight.Text
      Set TreeView1.DropHighlight = Nothing
```

```
      indrag = False
   End If
End Sub

Private Sub TreeView1_DragOver(Source As Control, x As Single, y As Single,
State As Integer)
   If indrag = True Then
      ' Set DropHighlight to the mouse's coordinates.
      Set TreeView1.DropHighlight = TreeView1.HitTest(x, y)
   End If
End Sub
```

EnsureVisible Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthEnsureVisibleC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthEnsureVisibleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthEnsureVisibleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthEnsureVisibleS"}

Ensures that a specified **ListItem** or **Node** object is visible. If necessary, this method expands **Node** objects and scrolls the **TreeView** control. The method only scrolls the **ListView** control.

Syntax

object.**EnsureVisible**

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

Return Values

Value	Description
True	The method returns **True** if the **ListView** or **TreeView** control must scroll and/or expand to expose the object.
False	The method returns **False** if no scrolling and/or expansion is required.

Remarks

Use the **EnsureVisible** method when you want a particular **Node** or **ListItem** object, which might be hidden deep in a **TreeView** or **ListView** control, to be visible.

EnsureVisible Method Example

This example adds many nodes to a **TreeView** control, and uses the **EnsureVisible** method to scroll and expand the tree. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and click the form to see the **TreeView** expand.

```
Private Sub Form_Load()
   Dim nodX As Node
   Dim i as Integer
   TreeView1.BorderStyle = FixedSingle ' Show borders.

   Set nodX = TreeView1.Nodes.Add(,,,"Root")   ' Add first node.
   For i = 1 to 15   ' Add 15 nodes
      Set nodX = TreeView1.Nodes.Add(i,,,"Node " & CStr(i))
   Next i

   Set nodX = TreeView1.Nodes.Add(,,,"Bottom") ' Add one with text.
   Set nodX = TreeView1.Nodes.Add(i,,,"Expanded") ' Add child to node.
   Set nodX = TreeView1.Nodes.Add(i+1,,,"Show me") ' Add a final child.
End Sub

Private Sub Form_Click()
   ' Tree will scroll and expand when you click the form.
   TreeView1.Nodes(TreeView1.Nodes.Count).EnsureVisible
End Sub
```

Expand Event (TreeView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtExpandC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtExpandX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtExpandTreeviewA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtExpandTreeviewS"}

Occurs when a **Node** object in a **TreeView** control is expanded, that is, when its child nodes become visible.

Syntax

Private Sub *object*_**Expand(ByVal** *node* **As Node)**

The Expand event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
node	A reference to the expanded **Node** object.

Remarks

The Expand event occurs after the Click and DblClick events.

The Expand event is generated in three ways: when the user double-clicks a **Node** object that has child nodes; when the **Expanded** property for a **Node** object is set to **True**; and when the plus/minus image is clicked. Use the Expand event to validate an object, as in the following example:

```
Private Sub TreeView1_Expand(ByVal Node As Node)
   If Node.Index <> 1 Then
      Node.Expanded = False  ' Prevent expand.
   End If
End Sub
```

Expand Event Example

This example adds several **Node** objects to a **TreeView** control. When a **Node** is expanded, the Expand event is generated, and information about the **Node** is displayed. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and expand the nodes.

```
Private Sub Form_Load()
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(, , "RP", "Root Parent")
   Set nodX = TreeView1.Nodes.Add("RP", tvwChild, "C1", "Child1")
   Set nodX = TreeView1.Nodes.Add("C1", tvwChild, "C2", "Child2")
   Set nodX = TreeView1.Nodes.Add("C2", tvwChild, "C3", " Child3")
   Set nodX = TreeView1.Nodes.Add("C2", tvwChild, "C4", " Child4")
   TreeView1.Style = tvwTreelinesPlusMinusText ' Style 6.
   TreeView1.LineStyle = tvwRootLines    ' Style 1
End Sub

Private Sub TreeView1_Expand(ByVal Node As Node)
   Select Case Node.Key Like "C*"
   Case Is = True
      MsgBox Node.Text & " is a child node."
   End Select
End Sub
```

Expanded Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproExpandedC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproExpandedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproExpandedNodeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproExpandedNodeS"}

Returns or sets a value that determines whether a **Node** object in a **TreeView** control is currently expanded or collapsed.

Syntax

object.**Expanded**[= *boolean*]

The **Expanded** 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 that specifies whether the node is expanded or collapsed.

The settings for *boolean* are:

Setting	Description
True	The **Node** is currently expanded.
False	The **Node** is currently collapsed.

Remarks

You can use the **Expanded** property to programmatically expand a **Node** object. The following code has the same effect as double-clicking the first **Node**:

```
TreeView1.Nodes(1).Expanded = True
```

When a **Node** object is expanded, the Expand event is generated.

If a **Node** object has no child nodes, the property value is ignored.

Expanded Property Example

This example adds several **Node** objects to a **TreeView** control. When you click the form, the **Expanded** property for each **Node** is set to **True**. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and click the form to expand all the **Node** objects.

```
Private Sub Form_Load()
   Dim nodX As Node
   Dim i as Integer
   TreeView1.BorderStyle = vbFixedSingle ' Show border.

   ' Create a root node.
   Set nodX = TreeView1.Nodes.Add(,,"root","Root")

   For i = 1 to 5 ' Add 5 child nodes.
      Set nodX = TreeView1.Nodes.Add(i,tvwChild,,"Node " & CStr(i))
   Next i
End Sub

Private Sub Form_Click()
   Dim I as Integer
   For I = 1 to TreeView1.Nodes.Count
      ' Expand all nodes.
      TreeView1.Nodes(i).Expanded = True
   Next I
End Sub
```

ExpandedImage Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproExpandedImageC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproExpandedImageX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproExpandedImageA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproExpandedImageS"}

Returns or sets the index or key value of a **ListImage** object in an associated **ImageList** control; the **ListImage** is displayed when a **Node** object is expanded.

Syntax

object.**ExpandedImage**[= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression or string expression that specifies, respectively, the index value or the key value of the image to be displayed.

Remarks

This property allows you to change the image associated with a **Node** object when the user double-clicks the node or when the **Node** object's **Expanded** property is set to **True**.

FirstSibling Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFirstSiblingC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFirstSiblingX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFirstSiblingA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFirstSiblingS"}

Returns a reference to the first sibling of a **Node** object in a **TreeView** control.

Syntax

object.**FirstSibling**

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

Remarks

The first sibling is the **Node** that appears in the first position in one level of a hierarchy of nodes. Which **Node** actually appears in the first position depends on whether or not the **Node** objects at that level are sorted, which is determined by the **Sorted** property.

The **Child**, **FirstSibling**, **LastSibling**, **Previous**, **Parent**, **Next**, and **Root** properties all return a reference to another **Node** object. Therefore you can simultaneously reference and perform operations on a **Node**, as follows:

```
With TreeView1.Nodes(x).FirstSibling
   .Text = "New text"
   .Key = "New key"
   .SelectedImage = 3
End With
```

You can also set an object variable to the referenced **Node**, as follows:

```
Dim NodFirstSib As Node
' Get a reference to the first sibling of Node x.
Set NodFirstSib = TreeView1.Nodes(x).FirstSibling
' Use this reference to perform operations on the first sibling Node.
With nodFirstSib
   .Text = "New text"    '  Change the text.
   .Key = "New key"  ' Change key.
   .SelectedImage = 3    ' Change SelectedImage.
End With
```

FirstSibling Property Example

This example adds several nodes to a **TreeView** control. The **FirstSibling** property, in conjunction with the **Next** property and the **LastSibling** property, is used to navigate through a clicked **Node** object's hierarchy . To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example and click the various nodes to see what is returned.

```
Private Sub Form_Load()
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,"dad","Mike") ' A first sibling.
   Set nodX = TreeView1.Nodes.Add(,,"mom","Carol")
   Set nodX = TreeView1.Nodes.Add(,,,"Alice")

   ' Marsha is the FirstSibling.
   Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Marsha")
   Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Jan")
   Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Cindy")
   nodX.EnsureVisible ' Show all nodes.

   ' Greg is the FirstSibling.
   Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Greg")
   Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Peter")
   Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Bobby")
   nodX.EnsureVisible ' Show all nodes.
End Sub

Private Sub TreeView1_NodeClick(ByVal Node As Node)
   Dim strText As String
   Dim n As Integer
   ' Set n to FirstSibling's index.
   n = Node.FirstSibling.Index
   ' Place FirstSibling's text & linefeed in string variable.
   strText = Node.FirstSibling.Text & vbLF
   While n <> Node.LastSibling.Index
   ' While n is not the index of the last sibling, go to the
   ' next sibling and place its text into the string variable.
      strText = strText & TreeView1.Nodes(n).Next.Text & vbLF
   ' Set n to the next node's index.
      n = TreeView1.Nodes(n).Next.Index
   Wend
   MsgBox strText ' Display results.
End Sub
```

FullPath Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFullPathC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFullPathX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFullPathA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFullPathS"}

Returns the fully qualified path of the referenced **Node** object in a **TreeView** control. When you assign this property to a string variable, the string is set to the **FullPath** of the node with the specified index.

Syntax

object.**FullPath**

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

Remarks

The fully qualified path is the concatenation of the text in the referenced **Node** object's **Text** property with the **Text** property values of all its ancestors. The value of the **PathSeparator** property determines the delimiter.

FullPath Property Example

This example adds several **Node** objects to a **TreeView** control and displays the fully qualified path of each when selected. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, then select a node and click the form to display the **Node** object's full path.

```
Private Sub Form_Load()
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,,"Root")
   Set nodX = TreeView1.Nodes.Add(1,tvwChild,,"Dir1")
   Set nodX = TreeView1.Nodes.Add(2,tvwChild,,"Dir2")
   Set nodX = TreeView1.Nodes.Add(3,tvwChild,,"Dir3")
   Set nodX = TreeView1.Nodes.Add(4,tvwChild,,"Dir4")
   nodX.EnsureVisible   ' Show all nodes.
   TreeView1.Style = tvwTreelinesText ' Style 4.
End Sub

Private Sub TreeView1_NodeClick(ByVal Node As Node)
   MsgBox Node.FullPath
End Sub
```

GetVisibleCount Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetVisibleCountC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetVisibleCountX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetVisibleCountA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetVisibleCountS"}

Returns the number of **Node** objects that fit in the internal area of a **TreeView** control.

Syntax

object.**GetVisibleCount**

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

Remarks

The number of **Node** objects is determined by how many lines can fit in a window. The total number of lines possible is determined by the height of the control and the **Size** property of the **Font** object. The count includes the partially visible item at the bottom of the list.

You can use the **GetVisibleCount** property to make sure that a minimum number of lines are visible so the user can accurately assess a hierarchy. If the minimum number of lines is not visible, you can reset the size of the **TreeView** using the **Height** property.

If a particular **Node** object must be visible, use the **EnsureVisible** method to scroll and expand the **TreeView** control.

GetVisibleCount Method Example

This example adds several **Node** objects to a **TreeView** control. When you click the form, the code uses the **GetVisibleCount** method to check how many lines are visible, and then enlarges the control to show all the objects. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and click the form to enlarge the control.

```
Private Sub Form_Load()
   Dim nodX As Node
   Dim i as Integer
   TreeView1.BorderStyle = 1 ' Show border.
   For i = 1 to 20
      Set nodX = TreeView1.Nodes.Add(,,,"Node " & CStr(i))
   Next I
   TreeView1.Height = 1500 ' TreeView is short, for comparison's sake.
End Sub

Private Sub Form_Click()
   While Treeview1.GetVisibleCount < 20
      ' Make the treeview larger.
      TreeView1.Height = TreeView1.Height + TreeView1.Font.Size
   Wend
End Sub
```

HitTest Method (ListView, TreeView Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthHitTestC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthHitTestX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthHitTestA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthHitTestS"}

Returns a reference to the **ListItem** object or **Node** object located at the coordinates of x and y. Most often used with drag-and-drop operations to determine if a drop target item is available at the present location.

Syntax

object.**HitTest (***x* **As Single,** *y* **As Single)**

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
x,y	Coordinates of a target object, which is either a **Node** object or a **ListItem** object.

Remarks

If no object exists at the specified coordinates, the **HitTest** method returns **Nothing**.

The **HitTest** method is most frequently used with the **DropHighlight** property to highlight an object as the mouse is dragged over it. The **DropHighlight** property requires a reference to a specific object that is to be highlighted. In order to determine that object, the **HitTest** method is used in combination with an event that returns x and y coordinates, such as the DragOver event, as follows:

```
Private Sub TreeView1_DragOver _
(Source As Control, X As Single, Y As Single, State As Integer)
   Set TreeView1.DropHighlight = TreeView1.HitTest(X,Y)
End Sub
```

Subsequently, you can use the **DropHighlight** property in the DragDrop event to return a reference to the last object the source control was dropped over, as shown in the following code:

```
Private Sub TreeView1_DragDrop _
(Source As Control, x As Single, y As Single)
   ' DropHighlight returns a reference to object drop occurred over.
   Me.Caption = TreeView1.DropHighlight.Text
   ' To release the DropHighlight reference, set it to Nothing.
   Set TreeView1.DropHighlight = Nothing
End Sub
```

Note in the preceding example that the **DropHighlight** property is set to **Nothing** after the procedure is completed. This must be done to release the highlight effect.

HitTest Method (ListView, TreeView Controls) Example

This example adds several **Node** objects to a **TreeView** control. After you select a **Node** object, you can drag it to any other **Node**. To try the example, place **TreeView** and **ImageList** controls on a form and paste the code into the form's Declaration section. Run the example and drag **Node** objects around to see the result.

```
' Declare global variables.
Dim indrag As Boolean ' Flag that signals a Drag Drop operation.
Dim nodX As Object ' Item that is being dragged.

Private Sub Form_Load()
   ' Load a bitmap into an Imagelist control.
   Dim imgX As ListImage
   Dim BitmapPath As String
   BitmapPath = "icons\mail\mail01a.ico"
   Set imgX = imagelist1.ListImages.Add(, , LoadPicture(BitmapPath))

   ' Initialize TreeView control and create several nodes.
   TreeView1.ImageList = imagelist1
   Dim nodX As Node  ' Create a tree.
   Set nodX = TreeView1.Nodes.Add(, , , "Parent1", 1)
   Set nodX = TreeView1.Nodes.Add(, , , "Parent2", 1)
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Child 1", 1)
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Child 2", 1)
   Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 3", 1)
   Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 4", 1)
   Set nodX = TreeView1.Nodes.Add(3, tvwChild, , "Child 5", 1)
   nodX.EnsureVisible ' Expand tree to show all nodes.
End Sub

Private Sub TreeView1_MouseDown_
(Button As Integer, Shift As Integer, x As Single, y As Single)
   Set nodX = TreeView1.SelectedItem ' Set the item being dragged.
End Sub

Private Sub TreeView1_MouseMove _
(Button As Integer, Shift As Integer, x As Single, y As Single)
   If Button = vbLeftButton Then ' Signal a Drag operation.
      indrag = True ' Set the flag to true.
      ' Set the drag icon with the CreateDragImage method.
      TreeView1.DragIcon = TreeView1.SelectedItem.CreateDragImage
      TreeView1.Drag vbBeginDrag ' Drag operation.
   End If
End Sub

Private Sub TreeView1_DragDrop_
(Source As Control, x As Single, y As Single)
   If TreeView1.DropHighlight Is Nothing Then
      Set TreeView1.DropHighlight = Nothing
      indrag = False
      Exit Sub
   Else
      If nodX = TreeView1.DropHighlight Then Exit Sub
      Cls
      Print nodX.Text & " dropped on " & TreeView1.DropHighlight.Text
      Set TreeView1.DropHighlight = Nothing
```

```
      indrag = False
   End If
End Sub

Private Sub TreeView1_DragOver(Source As Control, x As Single, y As Single,
State As Integer)
   If indrag = True Then
      ' Set DropHighlight to the mouse's coordinates.
      Set TreeView1.DropHighlight = TreeView1.HitTest(x, y)
   End If
End Sub
```

Indentation Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproIndentationC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIndentationX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIndentationA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIndentationS"}

Returns or sets the width of the indentation for a **TreeView** control. Each new child **Node** object is indented by this amount.

Syntax

object.**Indentation**[= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	An integer specifying the width that each child **Node** is indented.

Remarks

If you change the **Indentation** property at run time, the **TreeView** is redrawn to reflect the new width. The property value cannot be negative.

Indentation Property Example

This example adds several **Node** objects to a **TreeView** control, while the **Indentation** property is shown in the form's caption. An **OptionButton** control array provides alternate values for the **Indentation** width. To try the example, place a **TreeView** control and a control array of three **OptionButton** controls on a form, and paste the code into the form's Declarations section. Run the example, and click an **OptionButton** to change the **Indentation** property.

```
Private Sub Form_Load()
   ' Label OptionButton controls with Indentation choices.
   Option1(0).Caption = "250"
   Option1(1).Caption = "500"
   Option1(2).Caption = "1000"

   ' Select the first option, and set the indentation to 250 initially
   Option1(0).Value = True
   Treeview1.Indentation = 250

   Dim nodX As Node
   Dim i As Integer

   Set nodX = TreeView1.Nodes.Add(,,,CStr(1)) ' Add first node.

   For i = 1 To 6 ' Add 6 nodes.
      Set nodX = TreeView1.Nodes.Add(i,tvwChild,,CStr(i + 1))
   Next i

   nodX.EnsureVisible   ' Makes sure all nodes are visible.
   Form1.Caption = "Indentation = " & TreeView1.Indentation
End Sub

Private Sub Option1_Click(Index as Integer)    ' Change Indentation with
OptionButton value.
   TreeView1.Indentation = Val(Option1(Index).Caption)
   Form1.Caption = "Indentation = " & TreeView1.Indentation
End Sub
```

LabelEdit Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLabelEditC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLabelEditX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLabelEditA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLabelEditS"}

Returns or sets a value that determines if a user can edit labels of **ListItem** or **Node** objects in a **ListView** or **TreeView** control.

Syntax

object.**LabelEdit** [= *integer*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
integer	An integer that determines whether the label of a **Node** or **ListItem** object can be edited, as specified in Settings.

Settings

The settings for *integer* are:

Constant	Value	Description
ListView: **lvwAutomatic** TreeView: **tvwAutomatic**	0	(Default) Automatic. The BeforeLabelEdit event is generated when the user clicks the label of a selected node.
ListView: **lvwManual** TreeView: **tvwManual**	1	Manual. The BeforeLabelEdit event is generated only when the **StartLabelEdit** method is invoked.

Remarks

Label editing of an object is initiated when a selected object is clicked (if the **LabelEdit** property is set to Automatic). That is, the first click on an object will select it; a second (single) click on the object will initiate the label editing operation.

The **LabelEdit** property, in combination with the **StartLabelEdit** method, allows you to programmatically determine when and which labels can be edited. When the **LabelEdit** property is set to 1, no label can be edited unless the **StartLabelEdit** method is invoked. For example, the following code allows the user to edit a **Node** object's label by clicking a Command button:

```
Private Sub Command1_Click()
   ' Determine if the right Node is selected.
   If TreeView1.SelectedItem.Index = 1 Then
      TreeView1.StartLabelEdit ' Let user begin editing.
   End If
End Sub
```

LabelEdit Property Example

This example initiates label editing when you click the Command button. It allows a **Node** object to be edited unless it is a root **Node**. The **LabelEdit** property must be set to **Manual**. To try the example, place a **TreeView** control and a **CommandButton** on a form. Paste the code into the form's Declarations section. Run the example, select a node to edit, and press the **CommandButton**.

```
Private Sub Form_Load()
   Dim nodX As Node
   Dim i As Integer
   TreeView1.LabelEdit = tvwManual ' Set property to manual.
   Set nodX = TreeView1.Nodes.Add(,,," Node 1")      ' Add first node.

   For i = 1 to 5 ' Add 5 nodes.
      Set nodX = TreeView1.Nodes.Add(i,tvwChild,,"Node " & CStr(i + 1))
   Next I

   nodX.EnsureVisible   ' Show all nodes.
End Sub

Private Sub Command1_Click()
   ' Invoke the StartLabelEdit method on the selected node,
   ' which triggers the BeforeLabelEdit event.
   TreeView1.StartLabelEdit
End Sub

Private Sub TreeView_BeforeLabelEdit (Cancel as Integer)
   ' If the selected item is the root, then cancel the edit.
   If TreeView1.SelectedItem Is TreeView1.SelectedItem.Root Then
      Cancel = True
   End If
End Sub

Private Sub TreeView_AfterLabelEdit _
(Cancel As Integer, NewString As String)
   ' Assume user has entered some text and pressed the ENTER key.
   ' Any nonempty string will be valid.
   If Len(NewString) = 0 Then
      Cancel = True
   End If
End Sub
```

LastSibling Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLastSiblingC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLastSiblingX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLastSiblingA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLastSiblingS"}

Returns a reference to the last sibling of a **Node** object in a **TreeView** control.

Syntax

object.**LastSibling**

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

Remarks

The last sibling is the **Node** that appears in the last position in one level of a hierarchy of nodes. Which **Node** actually appears in the last position depends on whether or not the **Node** objects at that level are sorted, which is determined by the **Sorted** property. To sort the **Node** objects at one level, set the **Sorted** property of the **Parent** node to **True**. The following code demonstrates this:

```
Private Sub TreeView1_NodeClick(ByVal Node As Node)
   Node.Parent.Sorted = True
End Sub
```

The **Child**, **FirstSibling**, **LastSibling**, **Previous**, **Parent**, **Next**, and **Root** properties all return a reference to another **Node** object. Therefore, you can simultaneously reference and perform operations on a **Node**, as follows:

```
With TreeView1.Nodes(x).LastSibling
   .Text = "New text"
   .Key = "New key"
   .SelectedImage = 3
End With
```

You can also set an object variable to the referenced **Node**, as follows:

```
Dim NodLastSib As Node
' Get a reference to the last sibling of Node x.
Set NodLastSib = TreeView1.Nodes(x).LastSibling
' Use this reference to perform operations on the sibling Node.
With nodLastSib
   .Text = "New text"    '  Change the text.
   .Key = "New key"  ' Change key.
   .SelectedImage = 3    ' Change SelectedImage.
End With
```

LastSibling Property Example

This example adds several **Node** objects to a **TreeView** control. The **LastSibling** property, in conjunction with the **Next** property and the **FirstSibling** property, is used to navigate through a clicked **Node** object's hierarchy level. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and click the various nodes to see what is returned.

```
Private Sub Form_Load()
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,"dad","Mike")
   Szet nodX = TreeView1.Nodes.Add(,,"mom","Carol")
   ' Alice is the LastSibling.
   Set nodX = TreeView1.Nodes.Add(,,,"Alice")

   Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Marsha")
   Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Jan")
   ' Cindy is the LastSibling.
   Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Cindy")
   nodX.EnsureVisible ' Show all nodes.

   Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Greg")
   Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Peter")
   ' Bobby is the LastSibling.
   Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Bobby")
   nodX.EnsureVisible ' Show all nodes.
End Sub

Private Sub TreeView1_NodeClick(ByVal Node As Node)
   Dim strText As String
   Dim n As Integer
   ' Set n to FirstSibling's index.
   n = Node.FirstSibling.Index
   ' Place FirstSibling's text & linefeed in string variable.
   strText = Node.FirstSibling.Text & vbLF
   While n <> Node.LastSibling.Index
   ' While n is not the index of the last sibling, go to the
   ' next sibling and place its text into the string variable
      strText = strText & TreeView1.Nodes(n).Next.Text & vbLF
   ' Set n to the next node's index.
      n = TreeView1.Nodes(n).Next.Index
   Wend
   MsgBox strText ' Display results.
End Sub
```

LineStyle Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLineStyleC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLineStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLineStyleA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLineStyleS"}

Returns or sets the style of lines displayed between **Node** objects.

Syntax

object.**LineStyle** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A value or constant that specifies the line style as shown in Settings.

Settings

The settings for *number* are:

Constant	Value	Description
tvwTreeLines	0	(Default) Tree lines. Displays lines between **Node** siblings and their parent **Node**.
tvwRootLines	1	Root Lines. In addition to displaying lines between **Node** siblings and their parent **Node**, also displays lines between the root nodes.

Remarks

You must set the **Style** property to a style that includes tree lines.

LineStyle, Style Properties Example

This example adds several **Node** objects with images to a **TreeView** control. You can change the **LineStyle** and **Style** properties by selecting the alternate styles in two **OptionButton** control arrays. To try the example, place a **TreeView** control, an **ImageList** control, and two **OptionButton** control arrays (one with two buttons and one with eight) on a form, and paste the code into the form's Declarations section. Run the example, and click any **OptionButton** to change the **LineStyle** and **Style** properties.

```
Private Sub Form_Load()
   ' Add an image to the ImageList control.
   Dim imgX As ListImage
   Set imgX = ImageList1.ListImages. _
   Add(,,LoadPicture("bitmaps\outline\leaf.bmp"))

   TreeView1.ImageList = ImageList1 ' Initialize ImageList.

   ' Label OptionButton controls with line styles choices.
   Option1(0).Caption = "TreeLines"
   Option1(1).Caption = "RootLines"

   ' Select the first option, and set the LineStyle to TreeLines initially
   Option1(0).Value = True
   Treeview1.LineStyle = tvwTreeLines

   ' Label OptionButton controls with Style choices.
   Option2(0).Caption = "Text only"
   Option2(1).Caption = "Image & text"
   Option2(2).Caption = "Plus/minus & text"
   Option2(3).Caption = "Plus/minus, image & text"
   Option2(4).Caption = "Lines & text"
   Option2(5).Caption = "Lines, image & Text"
   Option2(6).Caption = "Lines, plus/minus & Text"
   Option2(7).Caption = "Lines, plus/minus, image & text"

   ' Select the last option, and set the initial Style
   Option2(7).Value = True
   Treeview1.Style = tvwTreelinesPlusMinusPictureText

   Dim nodX As Node
   Dim i as Integer
   ' Create root node.
   Set nodX = TreeView1.Nodes.Add(,,,"Node " & "1",1)

   For i = 1 to 5 ' Add 5 nodes.
      Set nodX = TreeView1.Nodes. _
      Add(i,tvwChild,,"Node " & CStr(i + 1),1)
   Next I
   nodX.EnsureVisible   ' Show all nodes.
End Sub

Private Sub Option1_Click(Index as Integer)
   ' Change line style from OptionButton.
   TreeView1.LineStyle = Index
End Sub

Private Sub Option2_Click(Index as Integer)
```

```
' Change Style with OptionButton.
   TreeView1.Style = Index
   Form1.Caption = "TreeView Style = " & Option2(Index).Caption
End Sub
```

Next Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproNextC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNextX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNextA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNextS"}

Returns a reference to the next sibling **Node** of a **TreeView** control's **Node** object.

Syntax

object.**Next**

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

Remarks

The **Child**, **FirstSibling**, **LastSibling**, **Previous**, **Parent**, **Next**, and **Root** properties all return a reference to another **Node** object. Therefore you can simultaneously reference and perform operations on a **Node**, as follows:

```
With TreeView1.Nodes(x).Child
   .Text = "New text"
   .Key = "New key"
   .SelectedImage = 3
End With
```

You can also set an object variable to the referenced **Node**, as follows:

```
Dim NodChild As Node
' Get a reference to the child of Node x.
Set NodChild = TreeView1.Nodes(x).Child
' Use this reference to perform operations on the child Node.
With nodChild
   .Text = "New text"    '  Change the text.
   .Key = "New key"  ' Change key.
   .SelectedImage = 3   ' Change SelectedImage.
End With
```

Next Property Example

This example adds several **Node** objects to a **TreeView** control. The **LastSibling** property, in conjunction with the **Next** property and the **FirstSibling** property, is used to navigate through a clicked **Node** object's hierarchy level. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and click the various nodes to see what is returned.

```
Private Sub Form_Load()
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,"dad","Mike")
   Set nodX = TreeView1.Nodes.Add(,,"mom","Carol")
   ' Alice is the LastSibling.
   Set nodX = TreeView1.Nodes.Add(,,,"Alice")

   Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Marsha")
   Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Jan")
   ' Cindy is the LastSibling.
   Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Cindy")
   nodX.EnsureVisible ' Show all nodes.

   Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Greg")
   Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Peter")
   ' Bobby is the LastSibling.
   Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Bobby")
   nodX.EnsureVisible ' Show all nodes.
End Sub

Private Sub TreeView1_NodeClick(ByVal Node As Node)
   Dim strText As String
   Dim n As Integer
   ' Set n to FirstSibling's index.
   n = Node.FirstSibling.Index
   ' Place FirstSibling's text & linefeed in string variable.
   strText = Node.FirstSibling.Text & vbLF
   ' While n is not the index of the last sibling, go to the
   ' next sibling and place its text into the string variable.
   While n <> Node.LastSibling.Index
      strText = strText & TreeView1.Nodes(n).Next.Text & vbLF
   ' Set n to the next node's index.
      n = TreeView1.Nodes(n).Next.Index
   Wend
   MsgBox strText ' Display results.
End Sub
```

NodeClick Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtNodeClickC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtNodeClickX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtNodeClickA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtNodeClickS"}

Occurs when a **Node** object is clicked.

Syntax

Private Sub *object*_**NodeClick(ByVal** *node* **As Node)**

The NodeClick event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
node	A reference to the clicked **Node** object.

Remarks

The standard Click event is generated when the user clicks any part of the **TreeView** control outside a node object. The NodeClick event is generated when the user clicks a particular **Node** object; the NodeClick event also returns a reference to a particular **Node** object which can be used to validate the **Node** before further action is taken.

The NodeClick event occurs before the standard Click event.

NodeClick Event Example

This example adds several **Node** objects to a **TreeView** control. When a **Node** is clicked, the NodeClick event is triggered and is used to get the **Node** object's index and text. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and click any **Node**.

```
Private Sub Form_Load()
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,"R","Root")
   nodX.Expanded = True
   Set nodX = TreeView1.Nodes.Add(,,"P","Parent")
   nodX.Expanded = True
   Set nodX = TreeView1.Nodes.Add("R",tvwChild,,"Child 1")
   Set nodX = TreeView1.Nodes.Add("R",tvwChild,,"Child 2")
   Set nodX = TreeView1.Nodes.Add("R",tvwChild,,"Child 3")
   Set nodX = TreeView1.Nodes.Add("P",tvwChild,,"Child 4")
   Set nodX = TreeView1.Nodes.Add("P",tvwChild,,"Child 5")
   Set nodX = TreeView1.Nodes.Add("P",tvwChild,,"Child 6")
End Sub

Private Sub TreeView1_NodeClick(ByVal Node As Node)
   Form1.Caption = "Index = " & Node.Index & " Text:" & Node.Text
End Sub
```

Nodes Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproNodesC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNodesX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNodesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNodesS"}

Returns a reference to a collection of **TreeView** control **Node** objects.

Syntax

object.**Nodes**

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

Remarks

You can manipulate **Node** objects using standard collection methods (for example, the **Add** and **Remove** methods). You can access each element in the collection by its index, or by a unique key that you store in the **Key** property.

Nodes Property Example

This example adds several **Node** objects to a **TreeView** control. When the form is clicked, a reference to each **Node** is used to display each **Node** object's text. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and click the form.

```
Private Sub Form_Load()
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,"R","Root")
   Set nodX = TreeView1.Nodes.Add("R", tvwChild,"C1","Child 1")
   Set nodX = TreeView1.Nodes.Add("R", tvwChild,"C2","Child 2")
   Set nodX = TreeView1.Nodes.Add("R", tvwChild,"C3","Child 3")
   Set nodX = TreeView1.Nodes.Add("R", tvwChild,"C4","Child 4")
   nodX.EnsureVisible
   TreeView1.Style = tvwTreelinesText ' Style 4.
   TreeView1.BorderStyle = vbFixedSingle
End Sub

Private Sub Form_Click()
   Dim i As Integer
   Dim strNodes As String
   For i = 1 To TreeView1.Nodes.Count
   strNodes = strNodes & TreeView1.Nodes(i).Index & " " & _
   "Key: " & TreeView1.Nodes(i).Key & " " & _
   "Text: " & TreeView1.Nodes(i).Text & vbLF
   Next i
   MsgBox strNodes
End Sub
```

Parent Property (Node Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproParentNodeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproParentNodeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproParentNodeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproParentNodeS"}

Returns or sets the parent object of a **Node** object. Available only at run time.

Syntax

object.**Parent**[= *node*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
node	A **Node** object that becomes the parent of the object.

Remarks

At run time, an error occurs if you set this property to an object that creates a loop. For example, you cannot set any **Node** to become a child **Node** of its own descendants.

The **Child**, **FirstSibling**, **LastSibling**, **Previous**, **Parent**, **Next**, and **Root** properties all return a reference to another **Node** object. Therefore, you can simultaneously reference and perform operations on a **Node**, as follows:

```
With TreeView1.Nodes(x).Parent
   .Text = "New text"
   .Key = "New key"
   .SelectedImage = 3
End With
```

You can also set an object variable to the referenced **Node**, as follows:

```
Dim NodParent As Node
' Get a reference to the parent of Node x.
Set NodParent = TreeView1.Nodes(x).Parent
' Use this reference to perform operations on the Parent Node.
With nodParent
   .Text = "New text"    '  Change the text.
   .Key = "New key"  ' Change key.
   .SelectedImage = 3    ' Change SelectedImage.
End With
```

Parent Property Example (Node Object)

This example adds several **Node** objects to a **TreeView** control. After you select a **Node** object, you can then click and drag it to any other **Node** to make it a child of the target **Node**. To try the example, place **TreeView** and **ImageList** controls on a form and paste the code into the form's Declaration section. Run the example and drag **Node** objects onto other **Node** objects to see the result.

```
' Declare global variables.
Dim indrag As Boolean ' Flag that signals a Drag Drop operation.
Dim nodX As Object ' Item that is being dragged.

Private Sub Form_Load()
   ' Load a bitmap into an Imagelist control.
   Dim imgX As ListImage
   Dim BitmapPath As String
   BitmapPath = "icons\mail\mail01a.ico"
   Set imgX = ImageList1.ListImages.Add(, , LoadPicture(BitmapPath))

   ' Initialize TreeView control and create several nodes.
   TreeView1.ImageList = ImageList1
   Dim nodX As Node     ' Create a tree.
   Set nodX = TreeView1.Nodes.Add(, , , "Parent1", 1)
   Set nodX = TreeView1.Nodes.Add(, , , "Parent2", 1)
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Child 1", 1)
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Child 2", 1)
   Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 3", 1)
   Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 4", 1)
   Set nodX = TreeView1.Nodes.Add(3, tvwChild, , "Child 5", 1)
   nodX.EnsureVisible ' Expand tree to show all nodes.
End Sub

Private Sub TreeView1_MouseDown(Button As Integer, Shift As Integer, x As
Single, y As Single)
   Set nodX = TreeView1.SelectedItem ' Set the item being dragged.
   Set TreeView1.DropHighlight = Nothing
End Sub

Private Sub TreeView1_MouseMove _
(Button As Integer, Shift As Integer, x As Single, y As Single)
   If Button = vbLeftButton Then ' Signal a Drag operation.
      indrag = True ' Set the flag to true.
      ' Set the drag icon with the CreateDragImage method.
      TreeView1.DragIcon = TreeView1.SelectedItem.CreateDragImage
      TreeView1.Drag vbBeginDrag ' Drag operation.
   End If
End Sub

Private Sub TreeView1_DragDrop(Source As Control, x As Single, y As Single)
   ' If user didn't move mouse or released it over an invalid area.
   If TreeView1.DropHighlight Is Nothing Then
      indrag = False
      Exit Sub
   Else
      ' Set dragged node's parent property to the target node.
      On Error GoTo checkerror ' To prevent circular errors.
      Set nodX.Parent = TreeView1.DropHighlight
      Cls
```

```
      Print TreeView1.DropHighlight.Text & _
      " is parent of " & nodX.Text
      ' Release the DropHighlight reference.
      Set TreeView1.DropHighlight = Nothing
      indrag = False
      Exit Sub ' Exit if no errors occured.
   End If

checkerror:
   ' Define constants to represent Visual Basic errors code.
   Const CircularError = 35614
   If Err.Number = CircularError Then
      Dim msg As String
      msg = "A node can't be made a child of its own children."
      ' Display the message box with an exclamation mark icon
      ' and with OK and Cancel buttons.
      If MsgBox(msg, vbExclamation & vbOKCancel) = vbOK Then
         ' Release the DropHighlight reference.
         indrag = False
         Set TreeView1.DropHighlight = Nothing
         Exit Sub
      End If
   End If
End Sub

Private Sub TreeView1_DragOver(Source As Control, x As Single, y As Single,
State As Integer)
   Set TreeView1.DropHighlight = TreeView1.HitTest(x, y)
End Sub
```

PathSeparator Property (TreeView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPathSeparatorC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPathSeparatorX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPathSeparatorA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPathSeparatorS"}

Returns or sets the delimiter character used for the path returned by the **FullPath** property.

Syntax

object.**PathSeparator** [= *string*]

The **PathSeparator** syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string that determines the **PathSeparator**, usually a single character.

Remarks

The default character is "\."

PathSeparator Property Example

This example adds several **Node** objects to a **TreeView** control, and uses an **OptionButton** control array to change the **PathSeparator** property. To try the example, place a **TreeView** control and an **OptionButton** control array on a form, and paste the code into the form's Declarations section. Run the example, select a **Node**, and click the form. Change the **PathSeparator** property value using the **OptionButtons**.

```
Private Sub Form_Load
   TreeView1.BorderStyle = vbFixedSingle   ' Show border.
   ' Label OptionButton controls with Style choices.
   Option1(0).Caption = "/"
   Option1(1).Caption = "-"
   Option1(2).Caption = ":"

   ' Select the last option, and set the initial Style
   Option2(1).Value = True
   Treeview1.PathSeparator = Option1(1).Caption

   Dim nodX As Node
   Dim i As Integer
   Set nodX = TreeView1.Nodes.Add(,,,CStr(1))  ' Add first node.

   For i = 1 to 5 ' Add other nodes.
      Set nodX = TreeView1.Nodes.Add(i,tvwChild,,CStr(i + 1))
   Next i

   nodX.EnsureVisible   ' Ensure all are visible.
End Sub

Private Sub Option1_Click(Index as Integer)
   ' Change the delimiter character.
   TreeView1.PathSeparator = Option1(Index).Caption
End Sub

Private Sub TreeView1_NodeClick(ByVal Node As Node)
   ' Show path in form's caption.
   Me.Caption = Node.FullPath
End Sub
```

Previous Property (Node Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPreviousNodeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPreviousNodeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPreviousNodeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPreviousNodeS"}

Returns a reference to the previous sibling of a **Node** object.

Syntax

object. **Previous**

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

Remarks

The **Child**, **FirstSibling**, **LastSibling**, **Previous**, **Parent**, **Next**, and **Root** properties all return a reference to another **Node** object. Therefore you can simultaneously reference and perform operations on a **Node**, as follows:

```
With TreeView1.Nodes(x).Previous
   .Text = "New text"
   .Key = "New key"
   .SelectedImage = 3
End With
```

You can also set an object variable to the referenced **Node**, as follows:

```
Dim NodPrevious As Node
' Get a reference to the node previouus to Node x.
Set NodChild = TreeView1.Nodes(x).Previous
' Use this reference to perform operations on the previous Node.
With nodPrevious
   .Text = "New text"    '  Change the text.
   .Key = "New key"  ' Change key.
   .SelectedImage = 3   ' Change SelectedImage.
End With
```

Previous Property Example

This example adds several nodes to a **TreeView** control. The **Previous** property, in conjunction with the **LastSibling** property and the **FirstSibling** property, is used to navigate through a clicked **Node** object's hierarchy level. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and click the various nodes to see what is returned.

```
Private Sub Form_Load()
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(, , "r", "Root")
   Set nodX = TreeView1.Nodes.Add(, , "p", "parent")

   Set nodX = TreeView1.Nodes.Add("r", tvwChild, , "Child 1")
   Set nodX = TreeView1.Nodes.Add("r", tvwChild, , "Child 2")
   Set nodX = TreeView1.Nodes.Add("r", tvwChild, , "Child 3")
   nodX.EnsureVisible ' Show all nodes.

   Set nodX = TreeView1.Nodes.Add("p", tvwChild, , "Child 4")
   Set nodX = TreeView1.Nodes.Add("p", tvwChild, , "Child 5")
   Set nodX = TreeView1.Nodes.Add("p", tvwChild, , "Child 6")
   nodX.EnsureVisible ' Show all nodes.
End Sub

Private Sub TreeView1_NodeClick(ByVal Node As Node)
   Dim strText As String
   Dim n As Integer
   ' Set n to LastSibling's index.
   n = Node.LastSibling.Index
   ' Place LastSibling's text & linefeed in string variable.
   strText = Node.LastSibling.Text & vbLF
   While n <> Node.FirstSibling.Index
      ' While n is not the index of the FirstSibling, go to the
      ' previous sibling and place its text into the string variable.
      strText = strText & TreeView1.Nodes(n).Previous.Text & vbLF
      ' Set n to the previous node's index.
      n = TreeView1.Nodes(n).Previous.Index
   Wend
   MsgBox strText ' Display results.
End Sub
```

Root Property (Node Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproRootC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRootX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRootA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRootS"}

Returns a reference to the root **Node** object of a selected **Node**.

Syntax

object.**Root**

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

Remarks

The **Child**, **FirstSibling**, **LastSibling**, **Previous**, **Parent**, **Next**, and **Root** properties all return a reference to another **Node** object. Therefore, you can simultaneously reference and perform operations on a **Node**, as follows:

```
With TreeView1.Nodes(x).Root
   .Text = "New text"
   .Key = "New key"
   .SelectedImage = 3
End With
```

You can also set an object variable to the referenced **Node**, as follows:

```
Dim NodRoot As Node
' Get a reference to the root of Node x.
Set NodRoot = TreeView1.Nodes(x).Root
' Use this reference to perform operations on the root Node.
With nodRoot
   .Text = "New text"    '  Change the text.
   .Key = "New key"  ' Change key.
   .SelectedImage = 3   ' Change SelectedImage.
End With
```

Root Property Example

This example adds several **Node** objects to a **TreeView** control. When you click a **Node**, the code navigates up the tree to the **Root** node, and displays the text of each **Parent** node. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and click a **Node**.

```
Private Sub Form_Load()
   Dim nodX As Node  ' Create a tree.
   Set nodX = TreeView1.Nodes.Add(,,"r", "Root")
   Set nodX = TreeView1.Nodes.Add(,,"p", "Parent")
   Set nodX = TreeView1.Nodes.Add("p",tvwChild,, "Child 1")
   nodX.EnsureVisible   ' Show all nodes.
   Set nodX = TreeView1.Nodes.Add("r",tvwChild,"C2", "Child 2")
   Set nodX = TreeView1.Nodes.Add("C2",tvwChild,"C3", "Child 3")
   Set nodX = TreeView1.Nodes.Add("C3",tvwChild,, "Child 4")
   Set nodX = TreeView1.Nodes.Add("C3",tvwChild,, "Child 5")
   nodX.EnsureVisible   ' Show all nodes.
End Sub

Private Sub TreeView1_NodeClick(ByVal Node As Node)
   Dim n As Integer
   Dim strParents As String  ' Variable for information.
   n = Node.Index ' Set n to index of clicked node.
   strParents = Node.Text & vbLF
   While n <> Node.Root.Index
      strParents = strParents & _
      TreeView1.Nodes(n).Parent.Text & vbLF
      ' Set n to index of next parent Node.
      n = TreeView1.Nodes(n).Parent.Index
   Wend
   MsgBox strParents
End Sub
```

Selected Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vastmSet;vbproBooksOnlineJumpTopic;vbproSelectedOLEC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelectedOLEX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelectedOLEA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelectedOLES"}

Returns or sets a value that determines if a **Node** or **Tab** object is selected. For a **ListItem** object, the **Selected** property does not set the **SelectedItem** property, and thus does not cause the object to be selected. It only returns a value indicating whether the **ListItem** object has already been selected by other means.

Syntax

object.**Selected** [= *boolean*]

The **Selected** 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 that determines if an object is selected.

Remarks

Use the **Selected** property to programmatically select a specific **Node** or **Tab** object. Once you have selected an object in this manner, you can perform various operations on it, such as setting properties and invoking methods.

To select a specific **Node** object, you must refer to it by the value of either its **Index** property or its **Key** property. The following example selects a specific **Node** object in a **TreeView** control:

```
Private Sub Command1_Click()
   TreeView1.Nodes(3).Selected = True ' Selects an object.
   ' Use the SelectedItem property to get a reference to the object.
   TreeView1.SelectedItem.Text = "Changed Text"
End Sub
```

In the **ListView** control, the **SelectedItem** property always refers to the first selected item. Therefore, if multiple items are selected, you must iterate through all of the items, checking each item's **Selected** property.

Note Instead of using the **Selected** property to programmatically select a **ListItem** object, use the **Set** statement with the **SelectedItem** property, as follows:

```
Set ListView1.SelectedItem = ListView1.ListItems(1)
```

Selected Property Example

This example adds several **Node** objects to a **TreeView** control. When a **Node** is selected, a reference to the selected **Node** is used to display its key. To try the example, place a **TreeView** control on a form, and paste the code into the form's Declarations section. Run the example, select a **Node**, and click the form.

```
Private Sub Form_Load()
   Dim nodX As Node  ' Create a tree.
   Set nodX = TreeView1.Nodes.Add(,,"r","Root")
   Set nodX = TreeView1.Nodes.Add(,,"p","Parent")
   Set nodX = TreeView1.Nodes.Add("p",tvwChild,,"Child 1")
   nodX.EnsureVisible   ' Show all nodes.
   Set nodX = TreeView1.Nodes.Add("r",tvwChild,"C2","Child 2")
   Set nodX = TreeView1.Nodes.Add("C2",tvwChild,"C3","Child 3")
   Set nodX = TreeView1.Nodes.Add("C3",tvwChild,,"Child 4")
   Set nodX = TreeView1.Nodes.Add("C3",tvwChild,,"Child 5")
   nodX.EnsureVisible   ' Show all nodes.
End Sub

Private Sub Form_Click()
   Dim intX As Integer
   On Error Resume Next ' If an integer isn't entered.
   intX = InputBox("Check Node",,TreeView1.SelectedItem.Index)
   If IsNumeric(intX) Then ' Ensure an integer was entered.
      If TreeView1.Nodes(intX).Selected = True Then
         MsgBox TreeView1.Nodes(intX).Text & " is selected."
      Else
         MsgBox "Not selected"
      End If
   End If
End Sub
```

The following example adds three **ListItem** objects to a **ListView** control. When you click the form, the code uses the **Selected** property to determine if a specific **ListItem** object is selected. To try the example, place a **ListView** control on a form and paste the code into the form's Declarations section. Run the example, select a **ListItem**, and click the form.

```
Private Sub Form_Load()
   Listview1.BorderStyle = vbFixedSingle  ' Show the border.
   Dim itmX As ListViewItem
   Set itmX = ListView1.ListItems.Add(,,"Item 1")
   Set itmX = ListView1.ListItems.Add(,,"Item 2")
   Set itmX = ListView1.ListItems.Add(,,"Item 3")
End Sub

Private Sub Form_Click()
   Dim intX As Integer
   On Error Resume Next ' If an integer isn't entered.
   intX = InputBox("Check Item", , Listview1.SelectedItem.Index)
   If IsNumeric(intX) Then ' Ensure an integer was entered.
      If ListView1.ListItems(intX).Selected = True Then
         MsgBox ListView1.ListItems(intX).Text & " is selected."
      Else
         MsgBox "Not selected"
      End If
   End If
End Sub
```

SelectedImage Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSelectedImageC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelectedImageX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelectedImageA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelectedImageS"}

Returns or sets the index or key value of a **ListImage** object in an associated **ImageList** control; the **ListImage** is displayed when a **Node** object is selected.

Syntax

object.**SelectedImage** [= *index*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer or unique string that identifies a **ListImage** object in an associated **ImageList** control. The integer is the value of the **ListImage** object's **Index** property; the string is the value of the **Key** property.

Remarks

If this property is set to **Null**, the mask of the default image specified by the **Image** property is used.

SelectedItem Property (ActiveX Controls)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vastmSet;vbproBooksOnlineJumpTopic;vbproSelectedItemC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelectedItemX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelecteditemA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelecteditemS"}

Returns a reference to a selected **ListItem**, **Node**, or **Tab** object.

Syntax

object.**SelectedItem**

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

Remarks

The **SelectedItem** property returns a reference to an object that can be used to set properties and invoke methods on the selected object. This property is typically used to return a reference to a **ListItem, Node**, or **Tab** or object that the user has clicked or selected. With this reference, you can validate an object before allowing any further action, as demonstrated in the following code:

```
Command1_Click()
   ' If the selected object is not the root, then remove the Node.
   If TreeView1.SelectedItem.Index <> 1 Then
      Treeview1.Nodes.Remove TreeView1.SelectedItem.Index
   End If
End Sub
```

To programmatically select a **ListItem** object, use the **Set** statement with the **SelectedItem** property, as follows:

```
Set ListView1.SelectedItem = ListView1.ListItems(1)
```

SelectedItem Property Example

This example adds several **Node** objects to a **TreeView** control. After you select a **Node**, click the form to see various properties of the **Node**. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, select a **Node**, and click the form.

```
Private Sub Form_Load()
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(, , "r", "Root")
   Set nodX = TreeView1.Nodes.Add("r", tvwChild, "c1", "Child 1")
   Set nodX = TreeView1.Nodes.Add("r", tvwChild, "c2", "Child 2")
   Set nodX = TreeView1.Nodes.Add("r", tvwChild, "c3", "Child 3")
   Set nodX = TreeView1.Nodes.Add("c3", tvwChild, "c4", "Child 4")
   Set nodX = TreeView1.Nodes.Add("c3", tvwChild, "c5", "Child 5")
   Set nodX = TreeView1.Nodes.Add("c5", tvwChild, "c6", "Child 6")
   Set nodX = TreeView1.Nodes.Add("c5", tvwChild, "c7", "Child 7")
   nodX.EnsureVisible
   TreeView1.BorderStyle = vbFixedSingle
End Sub

Private Sub Form_Click()
   Dim nodX As Node
   ' Set the variable to the SelectedItem.
   Set nodX = TreeView1.SelectedItem
   Dim strProps As String
   ' Retrieve properties of the node.
   strProps = "Text: " & nodX.Text & vbLF
   strProps = strProps & "Key: " & nodX.Key & vbLF
   On Error Resume Next ' Root node doesn't have a parent.
   strProps = strProps & "Parent: " & nodX.Parent.Text & vbLF
   strProps = strProps & "FirstSibling: " & _
   nodX.FirstSibling.Text & vbLF
   strProps = strProps & "LastSibling: " & _
   nodX.LastSibling.Text & vbLF
   strProps = strProps & "Next: " & nodX.Next.Text & vbLF

   MsgBox strProps
End Sub
```

Sorted Property (TreeView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSortedNodeC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSortedNodeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSortedNodeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSortedNodeS"}

- Returns or sets a value that determines whether the child nodes of a **Node** object are sorted alphabetically.
- Returns or sets a value that determines whether the root level nodes of a **TreeView** control are sorted alphabetically.

Syntax

object.**Sorted** [= *boolean*]

The **Sorted** 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 **Node** objects are sorted, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The **Node** objects are sorted alphabetically by their **Text** property. **Node** objects whose **Text** property begins with a number are sorted as strings, with the first digit determining the initial position in the sort, and subsequent digits determining sub-sorting.
False	The **Node** objects are not sorted.

Remarks

The **Sorted** property can be used in two ways: first, to sort the **Node** objects at the root (top) level of a **TreeView** control and, second, to sort the immediate children of any individual **Node** object. For example, the following code sorts the root nodes of a **TreeView** control:

```
Private Sub Command1_Click()
   TreeView1.Sorted = True ' Top level Node objects are sorted.
End Sub
```

The next example shows how to set the **Sorted** property for a **Node** object as it is created:

```
Private Sub Form_Load()
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,,"Parent Node")
   nodX.Sorted = True
End Sub
```

Setting the **Sorted** property to **True** sorts the current **Nodes** collection only. When you add new **Node** objects to a **TreeView** control, you must set the **Sorted** property to **True** again to sort the added **Node** objects.

Sorted Property (TreeView Control) Example

This example adds several **Node** objects to a tree. When you click a **Node**, you are asked if you want to sort the **Node**. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and click a **Node** to sort it.

```
Private Sub Form_Load()
   ' Create a tree with several unsorted Node objects.
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(, , , "Adam")
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, "z", "Zachariah")
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Noah")
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Abraham")
   Set nodX = TreeView1.Nodes.Add("z", tvwChild, , "Stan")
   Set nodX = TreeView1.Nodes.Add("z", tvwChild, , "Paul")
   Set nodX = TreeView1.Nodes.Add("z", tvwChild, "f", "Frances")
   Set nodX = TreeView1.Nodes.Add("f", tvwChild, , "Julie")
   Set nodX = TreeView1.Nodes.Add("f", tvwChild, "c", "Carol")
   Set nodX = TreeView1.Nodes.Add("f", tvwChild, , "Barry")
   Set nodX = TreeView1.Nodes.Add("c", tvwChild, , "Yale")
   Set nodX = TreeView1.Nodes.Add("c", tvwChild, , "Harvard")
   nodX.EnsureVisible
End Sub

Private Sub TreeView1_NodeClick(ByVal Node As Node)
   Dim answer As Integer
   ' Check if there are children nodes.
   If Node.Children > 1 Then ' There are more than one children nodes.
      answer = MsgBox("Sort this node?", vbYesNo)  ' Prompt user.
      If answer = vbYes Then ' User wants to sort.
         Node.Sorted = True
      End If
   End If
End Sub
```

Style Property (TreeView Control)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproStyleTreeviewC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproStyleTreeviewX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStyleTreeviewA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStyleTreeviewS"}

Returns or sets the type of graphics (images, text, plus/minus, and lines) and text that appear for each **Node** object in a **TreeView** control.

Syntax

object.**Style** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	An integer specifying the style of the graphics, as described in Settings.

Settings

The settings for *number* are:

Setting	Description
0	Text only.
1	Image and text.
2	Plus/minus and text.
3	Plus/minus, image, and text.
4	Lines and text.
5	Lines, image, and text.
6	Lines, plus/minus, and text.
7	(Default) Lines, plus/minus, image, and text.

Remarks

If the **Style** property is set to a value that includes lines, the **LineStyle** property determines the appearance of the lines. If the **Style** property is set to a value that does not include lines, the **LineStyle** property will be ignored.

StartLabelEdit Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthStartLabelEditC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthStartLabelEditX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthStartLabelEditA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthStartLabelEditS"}

Enables a user to edit a label.

Syntax

object.**StartLabelEdit**

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

Remarks

The **StartLabelEdit** method must be used to initiate a label editing operation when the **LabelEdit** property is set to 1 (Manual).

When the **StartLabelEdit** method is invoked upon an object, the BeforeLabelEdit event is also generated.

StartLabelEdit Method Example

This example adds several **Node** objects to a **TreeView** control. After a **Node** is selected, click the form to begin editing it. To try the example, place a **TreeView** control on a form, and paste the code into the form's Declarations section. Run the example, select a **Node**, and click the form.

```
Private Sub Form_Load
   Dim nodX As Node

   Set nodX = TreeView1.Nodes.Add(,,,"Da Vinci") ' Root
   Set nodX = TreeView1.Nodes.Add(1,tvwChild,,"Titian")
   Set nodX = TreeView1.Nodes.Add(1,tvwChild,,"Rembrandt")
   Set nodX = TreeView1.Nodes.Add(1,tvwChild,,"Goya")
   Set nodX = TreeView1.Nodes.Add(1,tvwChild,,"David")
   nodX.EnsureVisible ' Expand tree to see all nodes.
End Sub

Private Sub Form_Click()
   ' If selected Node isn't the Root node then allow edits.
   If TreeView1.SelectedItem.Index <> 1 Then
      TreeView1.StartLabelEdit
   End If
End Sub
```

Add

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

The **Add** keyword is used in these contexts:

Add Method (Buttons Collection)

Add Method (ColumnHeaders Collection)

Add Method (ListImages Collection)

Add Method (ListItems Collection)

Add Method (Nodes Collection)

Add Method (Panels Collection)

Add Method (Tabs Collection)

Index

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

The **Index** keyword is used in these contexts:

Index Property (Control Array)

Index Property (Custom Controls)

Sorted

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

The **Sorted** keyword is used in these contexts:

Sorted Property (ListView Control)

Sorted Property (TreeView Control)

Style

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

The **Style** keyword is used in these contexts:

Style Property (Button Object)

Style Property (Panel Object)

Style Property (StatusBar Control)

Style Property (TabStrip Control)

Style Property (TreeView Control)

Width

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

The **Width** keyword is used in these contexts:

Width Property (Panel Object)

Height, Width Properties (Custom Controls)

Alignment

The **Alignment** keyword is used in these contexts:

Alignment Property (ColumnHeader Object)

Alignment Property (Panel Object)

AutoSize

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

The **AutoSize** keyword is used in these contexts:

Autosize Property (Custom Controls)

Autosize Property (Panel Object)

Caption

The **Caption** keyword is used in these contexts:

Caption Property (Custom Controls)

Caption Property (Tab Object)

Change

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

The **Change** keyword is used in these contexts:

Change Event (Custom Controls)

Change Event (Toolbar Control)

ImageList

The **ImageList** keyword is used in these contexts:

ImageList Control

ImageList Property (Custom Controls)

Parent

The **Parent** keyword is used in these contexts:

Parent Property (Custom Controls)

Parent Property (Node Object)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtChangeEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtChangeEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtChangeEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtChangeEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtClickEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtClickEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtClickEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtClickEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtDblClickEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtDblClickEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtDblClickEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtDblClickEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtDragDropEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtDragDropEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtDragDropEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtDragDropEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtDragOverEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtDragOverEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtDragOverEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtDragOverEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtKeyDownEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtKeyDownEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtKeyDownEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtKeyDownEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtKeyPressEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtKeyPressEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtKeyPressEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtKeyPressEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtKeyUpEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtKeyUpEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtKeyUpEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtKeyUpEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtMouseDownEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtMouseDownEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtMouseDownEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtMouseDownEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtMouseMoveEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtMouseMoveEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtMouseMoveEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtMouseMoveEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtMouseUpEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtMouseUpEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtMouseUpEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtMouseUpEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLECompleteDragEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLECompleteDragEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLECompleteDragEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLECompleteDragEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEDragDropEventPHolderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEDragDropEventPHolderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEDragDropEventPHolderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEDragDropEventPHolderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEDragOverEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEDragOverEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEDragOverEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEDragOverEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEGiveFeedbackEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEGiveFeedbackEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEGiveFeedbackEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEGiveFeedbackEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLESetDataEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLESetDataEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLESetDataEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLESetDataEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEStartDragEventControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEStartDragEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEStartDragEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEStartDragEventControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRefreshMethodControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRefreshMethodControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRefreshMethodControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRefreshMethodControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthOLEDragMethodControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthOLEDragMethodControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthOLEDragMethodControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthOLEDragMethodControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAppearancePropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAppearancePropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAppearancePropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAppearancePropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBackColorPropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBackColorPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBackColorPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBackColorPropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproCaptionPropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCaptionPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCaptionPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCaptionPropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproEnabledPropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproEnabledPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproEnabledPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproEnabledPropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproFontPropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontPropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproForeColorPropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproForeColorPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproForeColorPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproForeColorPropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproIntegralHeightPropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIntegralHeightPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIntegralHeightPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIntegralHeightPropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproLockedPropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLockedPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLockedPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLockedPropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMouseIconPropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMouseIconPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMouseIconPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMouseIconPropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMousePointerPropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMousePointerPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMousePointerPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMousePointerPropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLEDragModePropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDragModePropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDragModePropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDragModePropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLEDropModePropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDropModePropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDropModePropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDropModePropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproSelLengthPropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelLengthPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelLengthPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelLengthPropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproSelStartPropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelStartPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelStartPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelStartPropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproSelTextPropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelTextPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelTextPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelTextPropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproStylePropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproStylePropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStylePropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStylePropertyControlsPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproTextPropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTextPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTextPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTextPropertyControlsPlaceholderS"}

Keyword Not Found

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

The keyword you've selected can't be found in Visual Basic Help. You may have misspelled the keyword, selected too much or too little text, or asked for help on a word that is not a valid Visual Basic keyword.

The easiest way to get help on a specific keyword is to position the insertion point anywhere within the keyword you want help on and press F1. You do not need to select the keyword. In fact, if you select only a portion of the keyword, or more than a single word, Help will not find what you're looking for.

To use the built-in Help search dialog box, press the "Help Topics" button on the toolbar.

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproLeftTopPropertiesPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLeftTopPropertiesPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHeightWidthPropertiesPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHeightWidthPropertiesPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproTagPropertyPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTagPropertyPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproToolTipTextPropertyPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproToolTipTextPropertyPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproCountPropertyPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCountPropertyPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproItemMethodPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproItemMethodPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHwndPropertyPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHwndPropertyPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproTabStopPropertyPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTabStopPropertyPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTabStopPropertyPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTabStopPropertyPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproVisiblePropertyPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproVisiblePropertyPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproVisiblePropertyPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproVisiblePropertyPlaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproPicturePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPicturePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPicturePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPicturePropertyplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMaxMinPropertiesplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMaxMinPropertiesplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMaxMinPropertiesplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMaxMinPropertiesplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthAddMethodplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAddMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAddMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAddMethodplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthClearMethodplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthClearMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthClearMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthClearMethodplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproContainerPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproContainerPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproContainerPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproContainerPropertyplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolControlsCollectionplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolControlsCollectionplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbcolControlsCollectionplaceholderP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbcolControlsCollectionplaceholderM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbcolControlsCollectionplaceholderE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcolControlsCollectionplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproCopiesPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCopiesPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCopiesPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCopiesPropertyplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjDataBindingObjectplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjDataBindingObjectplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjDataBindingObjectplaceholderP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjDataBindingObjectplaceholderM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjDataBindingObjectplaceholderE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjDataBindingObjectplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolDataBindingsCollectionplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolDataBindingsCollectionplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbcolDataBindingsCollectionplaceholderP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbcolDataBindingsCollectionplaceholderM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbcolDataBindingsCollectionplaceholderE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcolDataBindingsCollectionplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproFilenamePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFilenamePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFilenamePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFilenamePropertyplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproFontBoldFontItalicFontStrikeThruPropertiesplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontBoldFontItalicFontStrikeThruPropertiesplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontBoldFontItalicFontStrikeThruPropertiesplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontBoldFontItalicFontStrikeThruPropertiesplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproFontNamePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontNamePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontNamePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontNamePropertyplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproFontSizePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontSizePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontSizePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontSizePropertyplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetDataMethodplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetDataMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetDataMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetDataMethodplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetFormatMethodplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetFormatMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetFormatMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetFormatMethodplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHelpFilePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHelpFilePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHelpFilePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHelpFilePropertyplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproItemPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproItemPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproItemPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproItemPropertyplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRemoveMethodplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRemoveMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRemoveMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRemoveMethodplaceholderS"}

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSetDataMethodplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSetDataMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSetDataMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSetDataMethodplaceholderS"}

Align Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAlignPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAlignPropertyplaceholderS"}

DragIcon Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDragIconPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDragIconPropertyplaceholderS"}

DragMode Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDragModePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDragModePropertyplaceholderS"}

HelpContextID Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHelpContextIDPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHelpContextIDPropertyplaceholderS"}

TabIndex Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproTabIndexPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTabIndexPropertyplaceholderS"}

Alignment Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstAlignmentConstantsplaceholderC"}

Border Property Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstBorderPropertyConstantsplaceholderC"}

BorderStyle Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstBorderStyleConstantsplaceholderC"}

BorderStyle Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBorderStylePropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBorderStylePropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBorderStylePropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBorderStylePropertyActiveXControlsplaceholderS"}

Clear Method (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthClearMethodActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthClearMethodActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthClearMethodActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthClearMethodActiveXControlsplaceholderS"}

Clipboard Object Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstClipboardObjectConstantsplaceholderC"}

Color Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstColorConstantsplaceholderC"}

CommonDialog Control Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstCommonDialogControlConstantsplaceholderC"}

CommonDialog Error Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstCommonDialogErrorConstantsplaceholderC"}

Control Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstControlConstantsplaceholderC"}

DataBindings Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDataBindingsPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataBindingsPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataBindingsPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataBindingsPropertyplaceholderS"}

DDE Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstDDEConstantsplaceholderC"}

Drag-and-Drop Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstDragandDropConstantsplaceholderC"}

Drawing Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstDrawingConstantsplaceholderC"}

FetchVerbs Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthFetchVerbsMethodC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthFetchVerbsMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthFetchVerbsMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthFetchVerbsMethodS"}

Form Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstFormConstantsplaceholderC"}

Graphics Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstGraphicsConstantsplaceholderC"}

Grid Control Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstGridControlConstantsplaceholderC"}

Help Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstHelpConstantsplaceholderC"}

HideSelection Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHideSelectionPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHideSelectionPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHideSelectionPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHideSelectionPropertyActiveXControlsplaceholderS"}

HideSelection Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHideSelectionPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHideSelectionPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHideSelectionPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHideSelectionPropertyplaceholderS"}

Image Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproImagePropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproImagePropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproImagePropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproImagePropertyActiveXControlsplaceholderS"}

ImageList Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproImageListPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproImageListPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproImageListPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproImageListPropertyActiveXControlsplaceholderS"}

Index Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproIndexPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIndexPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIndexPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIndexPropertyActiveXControlsplaceholderS"}

Index Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproIndexPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIndexPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIndexPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIndexPropertyplaceholderS"}

Key Code Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstKeyCodeConstantsplaceholderC"}

Key Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproKeyPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproKeyPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproKeyPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproKeyPropertyActiveXControlsplaceholderS"}

Menu Accelerator Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstMenuAcceleratorConstantsplaceholderC"}

Menu Control Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstMenuControlConstantsplaceholderC"}

Miscellaneous Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstMiscellaneousConstantsplaceholderC"}

MousePointer Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstMousePointerConstantsplaceholderC"}

OLE Container Control Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstOLEContainerControlConstantsplaceholderC"}

Picture Object Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstPictureObjectConstantsplaceholderC"}

Printer Object Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstPrinterObjectConstantsplaceholderC"}

RasterOp Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstRasterOpConstantsplaceholderC"}

Remove Method (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRemoveMethodActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRemoveMethodActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRemoveMethodActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRemoveMethodActiveXControlsplaceholderS"}

ShowInTaskbar Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstShowInTaskbarPropertyplaceholderC"}

ShowTips Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproShowTipsPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproShowTipsPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproShowTipsPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproShowTipsPropertyActiveXControlsplaceholderS"}

ShowWhatsThis Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthShowWhatsThisMethodplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthShowWhatsThisMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthShowWhatsThisMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthShowWhatsThisMethodplaceholderS"}

Text Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproTextPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTextPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTextPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTextPropertyActiveXControlsplaceholderS"}

Value Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproValuePropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproValuePropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproValuePropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproValuePropertyActiveXControlsplaceholderS"}

Variant Type Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstVariantTypeConstantsplaceholderC"}

Visual Basic Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstVisualBasicConstantsplaceholderC"}

WhatsThisButton Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproWhatsThisButtonPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWhatsThisButtonPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWhatsThisButtonPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWhatsThisButtonPropertyplaceholderS"}

WhatsThisHelp Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproWhatsThisHelpPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWhatsThisHelpPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWhatsThisHelpPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWhatsThisHelpPropertyplaceholderS"}

WhatsThisHelpID Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproWhatsThisHelpIDPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWhatsThisHelpIDPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWhatsThisHelpIDPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWhatsThisHelpIDPropertyplaceholderS"}

WhatsThisMode Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthWhatsThisModeMethodplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthWhatsThisModeMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthWhatsThisModeMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthWhatsThisModeMethodplaceholderS"}

Windows 95 Control Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstWindows95ControlConstantsplaceholderC"}

ColIsVisible Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproColIsVisiblePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproColIsVisiblePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproColIsVisiblePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproColIsVisiblePropertyplaceholderS"}

ColPos Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproColPosPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproColPosPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproColPosPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproColPosPropertyplaceholderS"}

DataObject Object (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDataObjectObjectplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataObjectObjectplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataObjectObjectplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataObjectObjectplaceholderS"}

DataObjectFiles Collection (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolDataObjectFilesCollectionplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolDataObjectFilesCollectionplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbcolDataObjectFilesCollectionplaceholderP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbcolDataObjectFilesCollectionplaceholderM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbcolDataObjectFilesCollectionplaceholderE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcolDataObjectFilesCollectionplaceholderS"}

DataSource Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDataSourcePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataSourcePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataSourcePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataSourcePropertyplaceholderS"}

Drag Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthDragMethodplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthDragMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthDragMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthDragMethodplaceholderS"}

DrawMode Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDrawModePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDrawModePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDrawModePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDrawModePropertyplaceholderS"}

FixedAlignment Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproFixedAlignmentPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFixedAlignmentPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFixedAlignmentPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFixedAlignmentPropertyplaceholderS"}

GotFocus Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtGotFocusEventplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtGotFocusEventplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtGotFocusEventplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtGotFocusEventplaceholderS"}

GridLineWidth Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproGridLineWidthPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproGridLineWidthPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproGridLineWidthPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproGridLineWidthPropertyplaceholderS"}

Index Property (Control Array) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproIndexPropertyControlArrayplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIndexPropertyControlArrayplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIndexPropertyControlArrayplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIndexPropertyControlArrayplaceholderS"}

LostFocus Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtLostFocusEventplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtLostFocusEventplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtLostFocusEventplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtLostFocusEventplaceholderS"}

Move Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthMoveMethodplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthMoveMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthMoveMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthMoveMethodplaceholderS"}

Name Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproNamePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNamePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNamePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNamePropertyplaceholderS"}

Object Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproObjectPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproObjectPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproObjectPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproObjectPropertyplaceholderS"}

Parent Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproParentPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproParentPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproParentPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproParentPropertyplaceholderS"}

RowColChange Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtRowColChangeEventplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtRowColChangeEventplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtRowColChangeEventplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtRowColChangeEventplaceholderS"}

RowIsVisible Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRowIsVisiblePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRowIsVisiblePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRowIsVisiblePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRowIsVisiblePropertyplaceholderS"}

RowPos Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRowPosPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRowPosPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRowPosPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRowPosPropertyplaceholderS"}

SetFocus Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSetFocusMethodplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSetFocusMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSetFocusMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSetFocusMethodplaceholderS"}

ZOrder Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthZOrderMethodplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthZOrderMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthZOrderMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthZOrderMethodplaceholderS"}

Files Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthFilesMethodplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthFilesMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthFilesMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthFilesMethodplaceholderS"}