Outrider ActiveX Controls

Online Documentation

Last Updated: Friday, May 10, 1996

Welcome to the online documentation system for Outrider ActiveX Controls. This system is available as a standard Windows help file accessible from ActiveX Control environments (such as Microsoft Visual Basic®) and via the World Wide Web at http://www.outrider.com/activex/help.

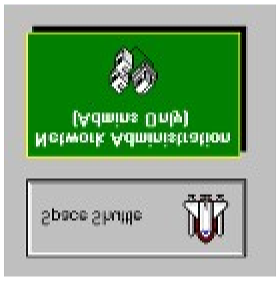
This documentation supports the following Outrider ActiveX Controls:

For comments on the documentation as well as technical support for the listed products, you may send e-mail to ***technicalsupport@outrider.com*** or fill out a Technical Support form at the Outrider Systems Web site listed above.

Outrider ButtonTool Control

[Properties](#) [Methods](#) [Events](#) [Outrider ActiveX Controls](#)

The Outrider ButtonTool Control is the latest version of one of the very first custom controls ever shipped. It provides an extremely-customizable command button that is capable of displaying text and graphics at the same time.



Included among the special effects are:

- >customizable beveling
- >precise caption placement
- >precise graphics placement
- >a wide array of “button down” effects
- >shadows
- >toggle modes

and so on.

The Outrider ButtonTool Control is flexible enough to jazz up any standalone application but it really shines when used on interactive Web pages capable of hosting ActiveX Controls.

Properties

The following methods are available to the Outrider ButtonTool Control:

BackColor
BevelLightColor
BevelShadeColor
BevelWidth
BorderColor
BorderWidth
ButtonDownEffect
Caption
CaptionAlignment
CaptionInset
CaptionShadow
CaptionShadowColor
Enabled
Font
ForeColor
hWnd
MouseIcon
MousePointer
PictureDown
PictureHeight
PictureUp
PictureWidth
PictureX
PictureY
RoundedCorners
ShadowColor
ShadowWidth
ShowFocusRect
ToggleMode
ToggleState

BackColor, ForeColor Properties

Returns or sets the background or foreground color of an Outrider ActiveX Control.

Syntax

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

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

Part	Description
object	An object expression that evaluates to an Outrider ActiveX Control.
color	A value or constant that determines the background or foreground color of a control, as described in Settings.

Settings

The settings for *color* are:

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

The default settings at design time are:

Property	Description
BackColor	The system default color specified by the constant **vbWindowBackground** (for the CheckList control) or **vbButtonFace** (for the ButtonTool and Enhanced SpinButton controls).
ForeColor	The system default color specified by the constant **vbWindowText** (for the CheckList control) or **vbButtonText** (for the ButtonTool and Enhanced SpinButton controls).

Remarks

For the Outrider Enhanced SpinButton Control, **BackColor** refers to that area of the spin control surrounding the up and down arrows. **ForeColor** refers to the color of the arrows. (The arrow colors are outlined in BorderColor.)
The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0; the lower 3 bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF). If the high byte isn't 0, Visual Basic

uses the system colors, as defined in the user's Control Panel settings and by constants listed in the Visual Basic (VB) object library in the Object Browser.

BevelLightColor, BevelShadeColor Properties

Returns or sets the colors used for beveling effects of an Outrider ActiveX Control.

Syntax

object.**BevelLightColor** [= *color*]
object.**BevelShadeColor** [= *color*]

The **BevelLightColor** and **BevelShadeColor** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an Outrider ActiveX Control.
color	A value or constant that determines the beveled edges of a control, as described in Settings.

Settings

The settings for *color* are:

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

The default settings at design time are:

Property	Description
BevelLightColor	Dark Gray (&H00808080&)
BevelForeColor	White (&H00FFFFFF&)

Remarks

For each control that supports beveling effects, **BevelLightColor** is used to paint the left and top edges of the control. **BevelShadeColor** is used to paint the right and bottom edges of the control.

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

BevelWidth Property

Returns or sets the bevel thickness of an Outrider ActiveX Control.

Syntax

object.**BevelWidth** [= *number*]

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

Part	Description
object	An object expression that evaluates to an Outrider ActiveX Control.
number	A numeric expression from 0 to 255, inclusive.

Remarks

The following table shows the effect of setting the **BevelWidth** property:

BevelWidth	Effect
0	No beveling effects are used.
1-255	Bevel effect is created with the width of each edge of the control equal to the **BevelWidth** value.

BorderColor Property

Returns or sets the color used to paint the border of an Outrider ActiveX Control.

Syntax

object.**BorderColor** [= *color*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider ActiveX Control.
color	A value or constant that determines the color of a control's border, as described in Settings.

Settings

The settings for *color* are:

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

The default setting for **BorderColor** at design time is black (&H0&).

Remarks

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

BorderWidth Property

Returns or sets the width of an Outrider ActiveX Control border.

Syntax

object.**BorderWidth** [= *number*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider ActiveX Control.
number	A numeric expression from 0 to 255, inclusive.

Remarks

The following table shows the effect of setting the **BorderWidth** property:

BorderWidth	**Effect**
0	No border is displayed.
1-255	Border is created with the width equal to the **BorderWidth** value.

Note that **BevelWidth** and **BorderWidth** are separate drawing properties and do not depend on the other's value.

ButtonDownEffect Property

Determines how an Outrider ButtonTool Control is displayed when it is clicked.

Syntax

object.**ButtonDownEffect** [= *number*]

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

Part	Description
object	An object expression that evaluates to an Outrider ActiveX Control.
number	A numeric expression from 0 to 7 inclusive. See Remarks below for information on each setting.

Remarks

Use this property to specify the way in which an Outrider ButtonTool Control will be displayed when it is clicked. The control also uses the value of this property to display itself when **ToggleMode** is True and **ToggleState** is True.

The following table shows the effect of setting the **ButtonDownEffect** property:

ButtonDownEffect	Effect
0	No effect - the button is displayed the same way whether it is in the “up” or “down” position.
1	Invert all colors.
2	Invert bevel colors only.
3	Temporarily halve the **BevelWidth** property.
4	Shift symbol down and to the right (similar to standard Windows command button behavior)
5	Button covers shadow completely
6	Button and shadow move 1/3 of the distance towards each other.
7	Button and shadow meet each other half way.

Note that certain values are dependent on whether you have specified a **BevelWidth** > 0 or **ShadowWidth** > 0.

Caption Property

Returns or sets the text displayed on an Outrider ButtonTool Control.

Syntax

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

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

Part	Description
object	An object expression that evaluates to an Outrider ButtonTool Control.
string	A string expression that evaluates to the text displayed as the caption.

Remarks

When you create a new control, the default caption is the default **Name** property setting. This default caption includes the object name and an integer, such as Osbt1.

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

CaptionAlignment Property

Returns or sets the position of the text on an Outrider ButtonTool Control.

Syntax

object.**CaptionAlignment** [= *number*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider ButtonTool Control.
string	A numeric expression in the range 0 to 8, inclusive. See Remarks for a description of each setting.

Remarks

The following table shows the effect of setting the **CaptionAlignment** property:

CaptionAlignment	**Position**
0	Aligned to Left and Top edge of button face
1	Left/Center
2	Left/Bottom
3	Center/Top
4	Center/Center (middle of button)
5	Center/Bottom
6	Right/Top
7	Right/Center
8	Right/Bottom

The text will be aligned to the edges of the button specified above. The text will also be placed at a certain distance from the edge according to the **CaptionInset** property.

CaptionInset Property

Returns or sets the distance from the edge of an Outrider ButtonTool Control face that its **Caption** will be displayed.

Syntax

object.**CaptionInset** [= *number*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider ButtonTool Control.
number	A numeric expression from 0 to 255, inclusive.

Remarks

For example, if the **CaptionInset** property is set to 4 pixels, and **CaptionAlignment** is set to 1 (Left/Center), the text displayed on the button will be positioned halfway down the button vertically, and 4 pixels from the left edge of the button face.

CaptionShadow Property

Determines whether or not the text on an Outrider ButtonTool Control is drawn in a 3-D fashion.

Syntax

object.**CaptionShadow** [= *boolean*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider ButtonTool Control.
boolean	A Boolean expression that specifies whether the control's text will be drawn in 3-D.

Settings

The settings for *boolean* are:

Setting	**Description**
True	The caption is actually drawn twice. First using the control's **ForeColor** property, then - offset by 1 pixel - its **CaptionShadowColor** property.
False	(Default) Caption is drawn without a 3-D effect.

CaptionShadowColor Property

Returns or sets the color used to paint the 3-D effect of the text displayed on an Outrider ButtonTool Control.

Syntax

object.**CaptionShadowColor** [= *color*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider ButtonTool Control.
color	A value or constant as described in Settings.

Settings

The settings for *color* are:

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

The default setting for **CaptionShadowColor** at design time is white (&HFFFFFF&).

Remarks

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

Enabled Property

Returns or sets a value that determines whether an Outrider ActiveX Control can respond to user-generated events.

Syntax

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

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

Part	**Description**
object	An object expression that evaluates to an Outrider ActiveX Control.
boolean	A Boolean expression that specifies whether the control can respond to user-generated events.

Settings

The settings for *boolean* are:

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

Remarks

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

Font Property

Returns or sets a Font object.

<u>Syntax</u>

object.**Font**

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

Part	**Description**
object	An object expression that evaluates to an Outrider ActiveX Control.

<u>Remarks</u>

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

```
osbtExit.Font.Bold = True
```

hWnd Property

Returns a handle to an Outrider ActiveX Control.

Syntax

object.**hWnd**

The object placeholder represents an object expression that evaluates to an Outrider ActiveX Control.

Remarks

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

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

MouseIcon Property

Sets a custom mouse icon.

Syntax

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

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

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

Remarks

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

You can use the **MouseIcon** property to load either cursor or icon files.   This provides your program with easy access to custom cursors of any size, with any desired hot spot location. The 32-bit version of Visual Basic does not load animated cursor (.ANI) files, even though 32-bit versions of Windows support these cursors.

MousePointer Property

Returns or sets a value indicating the type of mouse pointer displayed when the mouse is over an Outrider ActiveX Control at run time.

Syntax

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

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

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

Settings

The settings for *value* are:

Setting	Description
0	(Default) Shape determined by the object
1	Arrow
2	Cross (cross-hair pointer)
3	I-Beam
4	Icon (small square within a square)
5	Size (four-pointed arrow pointing north, south, east, and west)
6	Size NE SW (double arrow pointing northeast and southwest)
7	Size N S (double arrow pointing north and south)
8	Size NW SE (double arrow pointing northwest and southeast)
9	Size W E (double arrow pointing west and east)
10	Up Arrow
11	Hourglass (wait)
12	No Drop
13	Arrow and hourglass (Only available in 32-bit Windows)
14	Arrow and question mark (Only available in 32-bit Windows)
15	Size all (Only available in 32-bit Windows)
99	Custom icon specified by the **MouseIcon** property

Remarks

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

Note: When set for the **Screen** object, the mouse pointer may change across the entire screen. If your application doesn't call the **DoEvents** function and isn't a 32-bit application, it overrides all **MousePointer** settings for all controls and other applications. If your application calls **DoEvents**, the **MousePointer** property may temporarily change when over a custom control.

PictureDown, PictureUp Properties

Returns or sets a graphic to be displayed in an Outrider ButtonTool Control.

Syntax

object.**PictureDown** [= *picture*]
object.**PictureUp** [= *picture*]

The **PictureDown** and **PictureUp** property syntaxes have these parts:

Part	**Description**
object	An object expression that evaluates to an Outrider ButtonTool Control.
picture	A string expression specifying a file containing a graphic, as described in Settings.

Settings

The settings for *picture* are:

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

Remarks

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

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

PictureHeight, PictureWidth Properties

Determines how the graphics on an Outrider ButtonTool Control will be displayed.

Syntax

object.**PictureHeight** [= *number*]
object.**PictureWidth** [= *number*]

The **PictureHeight** and **PictureWidth** property syntaxes have these parts:

Part	**Description**
object	An object expression that evaluates to an Outrider ButtonTool Control.
number	If set to 0 (the default value), the picture will be scaled normally in that direction. For example, if you set **PictureHeight** to 0 and the graphic is inherently 64 pixels high, it will have the same height when it appears on the button. However, if you set *number* to any other value > 0, the graphic will be scaled to that height. For example, if you had set **PictureHeight** to 32, the graphic would appear half its normal height when displayed on the control.

PictureX, PictureY Properties

Determines how the graphics on an Outrider ButtonTool Control will be displayed.

Syntax

object.**PictureX** [= *number*]
object.**PictureY** [= *number*]

The **PictureX** and **PictureY** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an Outrider ButtonTool Control.
number	If set to -1 (the default value), the picture will be centered in the middle of the button. However, if you set *number* to any other value > 0, the graphic will be displayed *number* of pixels from either the left or top border (depending on whether you set **PictureX** or **PictureY**).

RoundedCorners Property

Returns or sets a value that determines whether an Outrider ButtonTool Control will show rounded corners.

Syntax

object.**RoundedCorners** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an Outrider ButtonTool Control.
boolean	A Boolean expression that specifies whether the control will show rounded corners.

Settings

The settings for *boolean* are:

Setting	Description
True	The control will have rounded corners.
False	(Default) The control will *not* have rounded corners.

ShadowColor Property

Returns or sets the color used to paint the shadow of an Outrider ActiveX Control.

Syntax

object.**ShadowColor** [= *color*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider ActiveX Control.
color	A value or constant that determines the color of a control’s shadow, as described in Settings.

Settings

The settings for *color* are:

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

The default setting for **ShadowColor** at design time is black (&H0&).

Remarks

That part of a control’s background that is “behind” the control but does not include the actual shadow is automatically painted using the control’s *parent container* BackColor property. This will generally be a small rectangle at the lower-left and upper-right hand corners of the control window.

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

ShadowWidth Property

Returns or sets the thickness of an Outrider ActiveX Control shadow.

Syntax

object.**ShadowWidth** [= *number*]

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

Part	Description
object	An object expression that evaluates to an Outrider ActiveX Control.
number	A numeric expression from 0 to 255, inclusive.

Remarks

The following table shows the effect of setting the **ShadowWidth** property:

ShadowWidth	Effect
0	No shadow is displayed.
1-255	Shadow is created with the width equal to the **ShadowWidth** value.

ShowFocusRect Property

Returns or sets a value that determines whether an Outrider ButtonTool Control will display the standard Windows focus rectangle when it has the focus.

Syntax

object.**ShowFocusRect** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an Outrider ButtonTool Control.
boolean	A Boolean expression that specifies whether the control will show a focus rectangle.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) The focus rectangle will be displayed.
False	The focus rectangle will *not* be displayed.

ToggleMode Property

Returns or sets a value that determines whether an Outrider ButtonTool Control behaves like a toggle switch.

Syntax

object.**ToggleMode** [= *boolean*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider ButtonTool Control.
boolean	A Boolean expression that specifies whether the control is in “toggle” mode.

Settings

The settings for *boolean* are:

Setting	**Description**
True	The control is in toggle mode.
False	(Default) The control is *not* in toggle mode.

Remarks

When in “toggle” mode, the control behaves as if it were a light switch. In other words, when you click the button, the button will stay “down” until you click it again.

ToggleState Property

Returns or sets a value that determines whether an Outrider ButtonTool Control is in the “up” or “down” state when **ToggleMode** is True.

Syntax

object.**ToggleState** [= *boolean*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider ButtonTool Control.
boolean	A Boolean expression that specifies whether the control is in the “up” or “down” state.

Settings

The settings for *boolean* are:

Setting	**Description**
True	The control is displayed in the “down” position.
False	(Default) The control is displayed in the “up” position.

Methods

The following properties are available to the Outrider ButtonTool Control:

DoClick
Refresh

DoClick Method

Automatically fires a **Click** event for an Outrider ActiveX Control.

Syntax

object.**DoClick**

The *object* placeholder represents an object expression that evaluates to an Outrider ActiveX Control.

Refresh Method

Forces a complete repaint of an Outrider ActiveX Control.

Syntax

object.**Refresh**

The *object* placeholder represents an object expression that evaluates to an Outrider ActiveX Control.

Remarks

Generally, painting a control is handled automatically while no events are occurring.   However, there may be situations where you want the control updated immediately. Use this method to force an immediate repainting of the control.

Events

The following events can be fired by the Outrider ButtonTool Control:

Click
DblClick
KeyDown
KeyPress
KeyUp
MouseDown
MouseEnter
MouseLeave
MouseMove
MouseUp

Click Event

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

Syntax

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

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

Part	**Description**
object	An object expression that evaluates to an Outrider ActiveX Control.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

Clicking a control generates **MouseDown** and **MouseUp** events in addition to the **Click** event. To distinguish between the left, right, and middle mouse buttons, use the **MouseDown** and **MouseUp** events.

DblClick Event

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

Syntax

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

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

Part	**Description**
object	An object expression that evaluates to an Outrider ActiveX Control.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

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

For those controls that receive mouse events, the events occur in this order: **MouseDown**, **MouseUp**, **Click**, **DblClick**, and **MouseUp**.

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

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

Key? Events

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

Syntax

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

The **Key?** event syntaxes have these parts:

Part	**Description**
object	An object expression that evaluates to an Outrider ActiveX Control.
index	Returns an integer that uniquely identifies a control if it's in a control array.
keycode	A key code, such as **vbKeyF1** (the F1 key) or **vbKeyHome** (the HOME key). To specify key codes, use the constants in the Visual Basic (VB) object library in the Object Browser.
shift	Returns an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released. A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.
keyascii	An integer that returns a standard numeric ANSI keycode. *Keyascii* is passed by reference; changing it sends a different character to the object. Changing *keyascii* to 0 cancels the keystroke so the object receives no character.

Remarks

Although the **KeyDown** and **KeyUp** events can apply to most keys, they're most often used for:

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

Use **KeyDown** and **KeyUp** event procedures if you need to respond to both the pressing and releasing of a key.You can convert the *keyascii* argument into a character by using the expression:

```
Chr(KeyAscii)
```

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

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

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

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

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

Constant	**Value**	**Description**
vbLeftButton	1	Left button is pressed.
vbRightButton	2	Right button is pressed.
vbMiddleButton	4	Middle button is pressed.
vbShiftMask	1	SHIFT key is pressed.
vbCtrlMask	2	CTRL key is pressed.
vbAltMask	4	ALT key is pressed.

Mouse? Events

These events occur when the user presses (**MouseDown**) or releases (**MouseUp**) a mouse button. **MouseMove** occurs when the user moves the mouse over an Outrider ActiveX Control. **MouseEnter** and **MouseLeave** occur when the mouse is moved into or out of the control area.

Syntax

Private Sub *object*_**MouseDown**([*index* **As Integer**,]*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single**)

Private Sub *object*_**MouseUp**([*index* **As Integer**,]*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single**)

Private Sub *object*_**MouseMove**([*index* **As Integer**,]*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single**)

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

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

The **Mouse?** event syntaxes have these parts:

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

Remarks

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

The **MouseEnter** event is fired when the mouse moves into the control window. The

MouseLeave event is fired when the mouse leaves the control window. Note that neither of these events are tied to mouse capture. Also, neither event has any parameters (other than the possible control *index* when the control is used in a control array).

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

Constant	**Value**	**Description**
vbLeftButton	1	Left button is pressed.
vbRightButton	2	Right button is pressed.
vbMiddleButton	4	Middle button is pressed.
vbShiftMask	1	SHIFT key is pressed.
vbCtrlMask	2	CTRL key is pressed.
vbAltMask	4	ALT key is pressed.

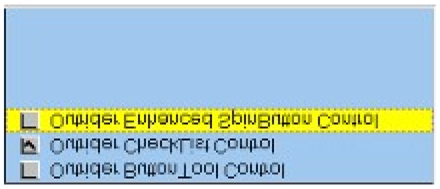
The constants then act as bit masks you can use to test for any combination of buttons without having to figure out the unique bit field value for each combination.

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

Outrider CheckList Control

Properties   Methods   Events   Outrider ActiveX Controls

The Outrider CheckList Control is designed to add extra functionality to the standard Windows list box. While it can function *just* as a list box, it also has the capability of displaying check boxes next to each item in the list.


Outrider Enhanced SpinButton Control
Outrider CheckList Control
Outrider ButtonTool Control

You will often see this type of control in various setup programs, where users are given the option of installing only the components they need. They simply check those items in the list they want installed. Now, with the Outrider CheckList Control, you have the ability to use this type of control in your application or Web page as well.

As you would expect, each visual aspect of the control is customizable. You can set the positions of the check box and item text columns in each row as well as specify the pictures you want use for the two states of an item - “checked” and “unchecked”. You can also allow users to check as many items as they wish, or restrict them to checking only one at a time. Finally, you can disable individual items in the list.

All of the methods you’ve used in the past to manipulate list boxes are found in the Outrider CheckList Control - **AddItem**, **Clear**, **List**, **ListIndex**, **RemoveItem**, etc.

Properties

The following properties are available to the Outrider CheckList Control:

Appearance
BackColor
CheckBoxLeft
CheckBoxes
Enabled
ExclusiveCheck
Font
ForeColor
hWnd
ItemChecked
ItemData
ItemEnabled
List
ListCount
ListIndex
MouseIcon
MousePointer
NewIndex
PictureChecked
PictureUnchecked
Sorted
TextLeft
TopIndex

Appearance Property

Returns or sets the paint style of an Outrider ActiveX Control.

Syntax

object.**Appearance** [= *number*]

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

Part	Description
object	An object expression that evaluates to an Outrider ActiveX Control.
number	A numeric expression that specifies the display style.

Settings

The settings for *number* are:

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

CheckBoxLeft Property

Returns or sets the distance, in pixels, from the left border of an Outrider CheckList Control where the check boxes will be displayed.

Syntax

object.**CheckBoxLeft** [= *number*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider CheckList Control.
number	A numeric expression from 0 to 32000, inclusive.

Remarks

Setting this property allows you to position the check box column anywhere in the row, even to the right of the item text, if you wish.

CheckBoxes Property

Returns or sets a value indicating whether check boxes will be displayed in an Outrider CheckList Control.

Syntax

object.**CheckBoxes** [= *boolean*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider CheckList Control.
boolean	A boolean expression specifying whether check boxes are displayed next to each item in the list. If set to False, no check boxes are displayed and the control functions like a standard Windows listbox.

ExclusiveCheck Property

Returns or sets a value indicating whether more than one item in an Outrider ActiveX Control can be checked.

Syntax

object.**ExclusiveCheck** [= *boolean*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider CheckList Control.
boolean	A boolean expression specifying whether multiple items can be checked. If True, only one item can be checked at a time.

ItemChecked Property

Returns or sets a value indicating whether an item in an Outrider ActiveX Control is checked.

Syntax

object.**ItemChecked**(*index*) [= *boolean*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider CheckList Control.
index	A numeric expression specifying the index of the current item.
boolean	A boolean expression specifying whether the item is checked.

Remarks

ItemChecked is used to access items in a list in the same manner that the **List** property is used.

ItemData

Returns or sets a specific number for each item in an Outrider CheckList Control.

Syntax

object.**ItemData**(*index*) [= *number*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider CheckList Control.
index	The number of a specific item in the list.
number	The number to be associated with the specified item.

Remarks

The **ItemData** property is an array of long integer values with the same number of items as a control's **List** property. You can use the numbers associated with each item to identify the items. For example, you can use an employee's identification number to identify each employee name in a control. When you fill the list, also fill the corresponding elements in the **ItemData** array with the employee numbers.

The **ItemData** property is often used as an index for an array of data structures associated with items in a control.

When you insert an item into a list with the **AddItem** method, an item is automatically inserted in the **ItemData** array as well. However, the value isn't reinitialized to zero; it retains the value that was in that position before you added the item to the list. When you use the **ItemData** property, be sure to set its value when adding new items to a list.

ItemEnabled Property

Returns or sets a value indicating whether an item in an Outrider ActiveX Control is enabled.

Syntax

object.**ItemEnabled**(*index*) [= *boolean*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider CheckList Control.
index	A numeric expression specifying the index of the current item.
boolean	A boolean expression specifying whether the item is enabled.

Remarks

ItemEnabled is used to access items in a list in the same manner that the **List** property is used.

List Property

Returns or sets the items contained in an Outrider CheckList Control list.   The list is a string array in which each element is a list item.

Syntax

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

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

Part	Description
object	An object expression that evaluates to an Outrider CheckList Control.
index	A numeric expression specifying the index of the current item.
string	A string expression specifying the list item.

Remarks

Use this property to access list items.
The index of the first item is 0 and the index of the last item is **ListCount** - 1.
The **List** property works in conjunction with the **ListCount** and **ListIndex** properties.
Enumerating a list from 0 to **ListCount** -1 returns all items in the list.
To specify items you want to display, use the **AddItem** method.   To remove items, use the **RemoveItem** method.   To keep items in alphabetic order, set the control's **Sorted** property to True before adding items to the list.

ListCount Property

Returns the number of items in the list portion of an Outrider CheckList Control.

Syntax

object.**ListCount**

The *object* placeholder represents an object expression that evaluates to an Outrider CheckList Control.

ListIndex Property

Returns or sets the index of the currently selected item in an Outrider CheckList Control.   Not available at design time.

Syntax

object.**ListIndex** [= *index*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider CheckList Control.
index	A numeric expression specifying the index of the current item.

Remarks

The expression List(Oscl.ListIndex) returns the string for the currently selected item.
The first item in the list is **ListIndex** = 0, and **ListCount** is always one more than the largest **ListIndex** value.

NewIndex Property

Returns the index of the item most recently added to an Outrider CheckList Control.

Syntax

object.**NewIndex**

The *object* placeholder represents an object expression that evaluates to an Outrider CheckList Control.

Remarks

You can use this property with sorted lists when you need a list of values that correspond to each item in the **ItemData** property array.   As you add an item in a sorted list, Visual Basic inserts the item in the list in alphabetic order.   This property tells you where the item was inserted so that you can insert a corresponding value in the **ItemData** property at the same index.

The **NewIndex** property returns **-**1 if there are no items in the list or if an item has been deleted since the last item was added.

PictureChecked, PictureUnchecked Properties

Returns or sets graphics to be displayed as check boxes in an Outrider CheckList Control.

Syntax

object.**PictureChecked** [= *picture*]
object.**PictureUnchecked** [= *picture*]

The **PictureChecked** and **PictureChecked** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an Outrider CheckList Control.
picture	A string expression specifying a file containing a graphic, as described in Settings.

Settings

The settings for *picture* are:

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

Remarks

Set these properties to determine how the check boxes will appear in both “checked” and “unchecked” states.

If a picture is not specified, the control will use it’s built-in 13x13 bitmap.

When specifying a graphic, you should use two graphics that are the same width and height.

Sorted Property

Returns a value indicating whether the elements of an Outrider CheckList Control are automatically sorted alphabetically.

Syntax

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

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

Part	**Description**
object	An object expression that evaluates to an Outrider CheckList Control.
boolean	Boolean value specifying whether the items in the list are to be sorted alphabetically (case-insensitive). The default is False.

Remarks

When this property is True, Visual Basic handles almost all necessary string processing to maintain alphabetic order, including changing the index numbers for items as required by the addition or removal of items.

Using the **AddItem** method to add an element to a specific location in the list may violate the sort order, and subsequent additions may not be correctly sorted.

TextLeft Property

Returns or sets the distance from the left border of an Outrider CheckList Control where the text of each item will be displayed.

Syntax

object.**TextLeft** [= *number*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider CheckList Control.
number	A numeric expression from 0 to 32000, inclusive.

Remarks

Setting this property allows you to position the text column anywhere in the row.

TopIndex Property

Returns or sets a value that specifies which item in an Outrider CheckList Control is displayed in the topmost position. Not available at design time.

Syntax

object.**TopIndex** [= *value*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider ActiveX Control.
value	The number of the list item that is displayed in the topmost position. The default is 0, or the first item in the list.

Remarks

Use this property to scroll through a CheckList control without selecting an item.

Methods

The following methods are available to the Outrider CheckList Control:

AddItem
Clear
Refresh
RemoveItem

AddItem Method

Adds an item to an Outrider CheckList Control.

Syntax

object.**AddItem** *item, index*

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

Part	**Description**
object	An object expression that evaluates to an Outrider CheckList Control.
item	String expression specifying the item to add to the list.
index	Optional. Integer specifying the position within the list where the new item. For the first item in the list, *index* = 0.

Remarks

If you supply a valid value for *index*, *item* is placed at that position within the list.   If *index* is omitted, *item* is added at the proper sorted position (if the **Sorted** property is set to True) or to the end of the list (if **Sorted** is set to False).

Clear Method

Clears the contents of an Outrider CheckList Control.

Syntax

object.**Clear**

The *object* placeholder represents an Outrider CheckList Control.

RemoveItem Method

Removes an item from an Outrider CheckList Control.

Syntax

object.**RemoveItem** *index*

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

Part	**Description**
object	An object expression that evaluates to an Outrider CheckList Control.
index	Integer representing the item to remove. For the first item in the control, *index* = 0.

Events

The following events can be fired by the Outrider CheckList Control:

Click
DblClick
ItemCheck
KeyDown
KeyPress
KeyUp
MouseDown
MouseEnter
MouseLeave
MouseMove
MouseUp

ItemCheck Event

This event occurs when the check box next to an item in an Outrider CheckList Control is clicked, either from user action or from code.

Syntax

Private Sub *object*_**ItemCheck**([*index* **As Integer**,] **ByVal** *ItemIndex* **As Integer**, **ByVal** *Checked* **As Boolean**)

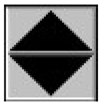
The **ItemCheck** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an Outrider CheckList Control.
index	Returns an integer that uniquely identifies a control if it's in a control array.
ItemIndex	Returns an integer that identifies the item in the list that is being clicked.
Checked	Returns a boolean value specifying whether the item is being ‘‘checked’’ or ‘‘unchecked’’.

Outrider Enhanced SpinButton Control

Properties Methods Events Outrider ActiveX Controls

The Outrider Enhanced SpinButton Control is an enhanced version of the Outrider SpinButton Control that has shipped with Microsoft Visual Basic since 1991.



The SpinButton control is used most often with another control to increment and decrement a value in that control, such as a date or time field. You can also use it to scroll back and forth through a range of values or a list of items.

At run time, when the user clicks the up (or right) arrow of the spin control, SpinUp events are generated repeatedly until the user releases the mouse. Likewise, when the user clicks the down (or left) arrow, SpinDown events are generated until the user releases the mouse. When using this control, you write code for the SpinUp and SpinDown events that increments or decrements the desired values.

The Delay property determines how often the SpinUp and SpinDown events are generated.

You can customize the appearance of the control by setting over a dozen display properties, including whether the arrows on the control are displayed vertically or horizontally (see SpinOrientation).

Among the *new* features in this version are:

- the ability to accelerate the spin control the longer the user leaves the mouse button pressed (see AcceleratorInterval, AcceleratorMinimumDelay, and AcceleratorRate),
- the ability to enable or disable the up and/or down arrows through code (although the control can also do it for you automatically) (see UpArrowEnabled and DownArrowEnabled), and
- an internal counter that is automatically incremented and decremented. You can use this value for whatever purposes your application requires. You can also set minimum and maximum values (see Value, MaximumValue, and MinimumValue).

Properties

The following properties are available to the Outrider Enhanced SpinButton Control:

AcceleratorInterval
AcceleratorMinimumDelay
AcceleratorRate
BackColor
BevelLightColor
BevelShadeColor
BevelWidth
BorderColor
BorderWidth
Delay
DownArrowEnabled
Enabled
ForeColor
hWnd
MaximumValue
MinimumValue
MouseIcon
MousePointer
ShadowColor
ShadowWidth
SpinOrientation
UpArrowEnabled
Value

AcceleratorInterval Property

Returns or sets the number of Spin events that are fired before the Outrider Enhanced SpinButton Control speeds up. Controls the acceleration of the spin control.

Syntax

object.**AcceleratorInterval** [= *number*]

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

Part	Description
object	An object expression that evaluates to an Outrider Enhanced SpinButton Control.
number	A numeric expression from 1 to 32767, inclusive.

Remarks

The AcceleratorInterval property is used in conjunction with the AcceleratorMinimumDelay and AcceleratorRate properties.

After a number of Spin events equal to the AcceleratorInterval value are fired, the Delay property will be decreased by a certain percentage equal to the value of the AcceleratorRate property, *but* only down to a value equal to the value of AcceleratorMinimumDelay property.

For example, assume that AcceleratorInterval is 3, AcceleratorMinimumDelay is 50, and AcceleratorRate is 25. After every 3rd Spin event, the Delay property is decreased 25%, down to a minimum of 50 milliseconds.

Note: The Delay property only changes while the mouse button is down (i.e. - the button is “spinning”). When the mouse button is released, the Delay property is reset to it’s original value.

AcceleratorMinimumDelay Property

Returns or sets the minimum delay value, in milliseconds, that the Outrider Enhanced SpinButton Control will accelerate to.

Syntax

object.AcceleratorMinimumDelay [= *number*]

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

Part	Description
object	An object expression that evaluates to an Outrider Enhanced SpinButton Control.
number	A numeric expression from 1 to 32767, inclusive.

Remarks

The AcceleratorMinimumDelay property is used in conjunction with the AcceleratorInterval and AcceleratorRate properties.

After a number of Spin events equal to the AcceleratorInterval value are fired, the Delay property will be decreased by a certain percentage equal to the value of the AcceleratorRate property, *but* only down to a value equal to the value of AcceleratorMinimumDelay property.

For example, assume that AcceleratorInterval is 3, AcceleratorMinimumDelay is 50, and AcceleratorRate is 25. After every 3rd Spin event, the Delay property is decreased 25%, down to a minimum of 50 milliseconds.

Note: The Delay property only changes while the mouse button is down (i.e. - the button is “spinning”). When the mouse button is released, the Delay property is reset to it’s original value.

AcceleratorRate Property

Returns or sets the rate of acceleration for an Outrider Enhanced SpinButton Control.

Syntax

object.**AcceleratorRate**[= *number*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider Enhanced SpinButton Control.
number	A numeric expression from 0 to 99, inclusive.

Remarks

The AcceleratorRate property is used in conjunction with the AcceleratorInterval and AcceleratorMinimumDelay properties.

After a number of Spin events equal to the AcceleratorInterval value are fired, the Delay property will be decreased by a certain percentage equal to the value of the AcceleratorRate property, *but* only down to a value equal to the value of AcceleratorMinimumDelay property.

For example, assume that AcceleratorInterval is 3, AcceleratorMinimumDelay is 50, and AcceleratorRate is 25. After every 3rd Spin event, the Delay property is decreased 25%, down to a minimum of 50 milliseconds.

The default value for AcceleratorRate is 0, which means the control will not accelerate at all.

Note: The Delay property only changes while the mouse button is down (i.e. - the button is "spinning"). When the mouse button is released, the Delay property is reset to it's original value.

Delay Property

Returns or sets the delay between SpinUp or SpinDown events. The **Delay** property slows the number of Spin events generated when the user clicks one of the arrows in an Outrider Enhanced SpinButton Control and then continues to hold down the button.

Syntax

object.**Delay** [= *number*]

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

Part	Description
object	An object expression that evaluates to an Outrider Enhanced SpinButton Control.
number	A numeric expression from 0 to 32767, inclusive.

Remarks

The following table lists the **Delay** property settings:

Setting	Description
250	(Default) 250 milliseconds, 1/4 of a second.
0-32767	Milliseconds delay between events.

DownArrowEnabled, UpArrowEnabled Properties

Returns or sets a value that determines whether or not one of the arrows in an Outrider Enhanced SpinButton Control is enabled.

Syntax

object.**DownArrowEnabled** [= *boolean*]
object.**UpArrowEnabled** [= *boolean*]

The **DownArrowEnabled** and **UpArrowEnabled** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an Outrider Enhanced SpinButton Control.
boolean	A Boolean expression that specifies whether the specified arrow is enabled.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) The specified arrow is enabled.
False	The specified arrow is disabled.

Remarks

When the up or down arrows are disabled, the arrows not drawn on the control.

Note also that while you can enable/disable the arrows yourself, the control will *always* disable the up arrow if Value is equal to MaximumValue, or the down arrow if Value is equal to MinimumValue.

MaximumValue, MinimumValue Properties

Returns or sets the maximum and minimum values for the Value property of an Outrider Enhanced SpinButton Control.

Syntax

object.**MaximumValue** [= *number*]
object.**MinimumValue** [= *number*]

The **MaximumValue** and **MinimumValue** property syntaxes have these parts:

Part	**Description**
object	An object expression that evaluates to an Outrider Enhanced SpinButton Control.
number	A long value between -2,147,483,648 and 2,147,483,647.

Remarks

The default values for MinimumValue and MaximumValue are -2,147,483,648 and 2,147,483,647, respectively. This allows you to implement the spin control without worrying about the value properties. (It's not likely application users will spin up or down to the maximum or minimum values if you leave these settings to their default values.)

If the Value property reaches the specified MaximumValue, the up arrow will be automatically disabled until Value drops below MaximumValue. If the Value property reaches the specified MinimumValue, the down arrow will be automatically disabled until Value rises above MaximumValue.

SpinOrientation Property

Sets the direction of the spin control arrows in an Outrider Enhanced SpinButton Control.

Syntax

object.**SpinOrientation** [= *number*]

Part	Description
object	An object expression that evaluates to an Outrider Enhanced SpinButton Control.
number	A numeric expression from 0 to 1, inclusive.

Settings

The following table lists the **SpinOrientation** property settings for an Outrider Enhanced SpinButton Control:

Setting	Description
0	(Default) Vertical: up and down arrows.
1	Horizontal: left and right arrows.

Remarks

You can also use the built-in constants **osVertical** and **osHorizontal**.

Value Property

Returns or sets the automatically incremented and decremented value of an Outrider Enhanced SpinButton Control.

Syntax

object. **Value** [= *number*]

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

Part	**Description**
object	An object expression that evaluates to an Outrider Enhanced SpinButton Control.
number	A long value between -2,147,483,648 and 2,147,483,647.

Remarks

The default **Value** is 0. You can use this property in whatever manner your application requires. **Value** is automatically incremented when the user clicks the up arrow and decremented when the down arrow is clicked.

Value will never be incremented or decremented outside the range MinimumValue to MaximumValue, inclusive. If either limit is reached, the appropriate arrow on the control will be disabled (i.e. - made invisible).

For example, if you have set MinimumValue to -10 and MaximumValue to 10, the up arrow will be disabled if Value is 10. The down arrow would be disabled if **Value** was decremented (or set in code) to -10.

Methods

The following methods are available to the Outrider Enhanced SpinButton Control:

Refresh

Events

The following events can be fired by the Outrider Enhanced SpinButton Control:

MouseDown
MouseEnter
MouseLeave
MouseMove
MouseUp
SpinDown
SpinUp

SpinDown, SpinUp Events

Occurs when the user clicks one of the arrows of an Outrider Enhanced SpinButton Control.

Syntax

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

The *object* placeholder represents an object expression that evaluates to an Outrider Enhanced SpinButton Control.

Remarks

SpinUp is generated when the up or the right arrow is clicked. **SpinDown** is generated by clicking the down or the left arrow. The arrows can be clicked once to send a single Spin event, or the left mouse button can be held down to generate multiple events.

Holding down the mouse button allows the user to cycle through a range of values. The Delay property slows the rate of cycling.

If you change the value or contents of a control in response to a Spin event, you must also call that control's Refresh method to insure that the updated value is displayed. Otherwise, the contents of that control may not be visible until the user releases the mouse button.