Visual Basic: Multimedia MCI Control

Visual Studio 6.0

Wait Property (Multimedia MCI Control)

See Also Example Applies To

Determines whether the **Multimedia MCI** control waits for the next **MCI** command to complete before returning control to the application. This property is not available at design time.

Syntax

[*form.*]*MMControl.***Wait**[= {**True | False**}]

Remarks

The following table lists the **Wait** property settings for the **Multimedia MCI** control.

Setting	Description
False	**Multimedia MCI** does not wait until the MCI command completes before returning control to the application.
True	**Multimedia MCI** waits until the next MCI command completes before returning control to the application.

The value assigned to this property is used only with the next **MCI** command. Subsequent **MCI** commands ignore the **Wait** property until it is assigned another (different or identical) value.

Data Type

Integer (Boolean)

Visual Basic: Multimedia MCI Control

Examples (Multimedia MCI Control)

Visual Basic Example

The following example illustrates the procedure used to open an MCI device with a compatible data file. By placing this code in the Form_Load procedure, your application can use the **Multimedia MCI** control "as is" to play, record, and rewind the file Gong.wav. To try this example, first create a form with a **Multimedia MCI** control.

```
Private Sub Form_Load ()
   ' Set properties needed by MCI to open.
   MMControl1.Notify = FALSE
   MMControl1.Wait = TRUE
   MMControl1.Shareable = FALSE
   MMControl1.DeviceType = "WaveAudio"
   MMControl1.FileName = "C:\WINDOWS\MMDATA\GONG.WAV"

   ' Open the MCI WaveAudio device.
   MMControl1.Command = "Open"
End Sub
```

To properly manage multimedia resources, you should close those MCI devices that are open before exiting your application. You can place the following statement in the Form_Unload procedure to close an open MCI device before exiting from the form containing the **Multimedia MCI** control.

```
Private Sub Form_Unload (Cancel As Integer)
   MMControl1.Command = "Close"
End Sub
```

Visual Studio 6.0

Visual Basic: MSChart Control

Wall Property

See Also Example Applies To

Returns a reference to a **Wall** object that describes the planar area depicting the y axes on a three-dimensional chart.

Syntax

object.**Wall**

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

Visual Basic: Windows Controls

Visual Studio 6.0

Week Property

See Also Example Applies To

Returns or sets a value that specifies the current week number.

Syntax

object.**Week** [= *number*]

The **Week** 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 that evaluates to an integer indicating the week number.

Remarks

The **Week** property can be set to any integer from 1 to 52.

Visual Basic Reference

Visual Studio 6.0

Weight Property

See Also Example Applies To

Returns or sets the weight of the characters that make up a **Font** object. The weight refers to the thickness of the characters, or the boldness factor. The higher the value, the bolder the character.

Syntax

object.**Weight** [= *number*]

The **Weight** 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 specifying the weight of the font.

Remarks

The **Font** object isn't directly available at design time. You set the **Weight** property of the **Font** object by selecting a control's **Font** property in the Properties window and clicking the Properties button. You implicitly set the **Weight** property by selecting an item from the Font Style box in the Font dialog box. The Regular and Italic settings have a **Weight** value of 400 (the default), and the Bold and Bold Italic settings have a **Weight** value of 700. At run time, however, you set **Weight** directly by specifying its setting for the **Font** object.

If you set a **Font** object's **Weight** to a value other than 400 or 700 at run time, Visual Basic converts your value to either 400 or 700, depending on which value is closest to the value you set. The precise ranges are: **Weight** > 400 and < 551 converts to 400; **Weight** > 550 converts to 700.

Visual Basic Reference

Bold, Italic, Size, StrikeThrough, Underline, Weight Properties Example

This example prints text on a form with each mouse click. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form twice.

```
Private Sub Form_Click ()
   Font.Bold = Not Font.Bold   ' Toggle bold.
   Font.StrikeThrough = Not Font.StrikeThrough ' Toggle strikethrough.
   Font.Italic = Not Font.Italic   ' Toggle italic.
   Font.Underline = Not Font.Underline   ' Toggle underline.
   Font.Size = 16   ' Set Size property.
   If Font.Bold Then
      Print "Font weight is " & Font.Weight & " (bold)."
   Else
      Print "Font weight is " & Font.Weight & " (not bold)."
   End If
End Sub
```

Visual Studio 6.0

Visual Basic: MSChart Control

Weighting Property

See Also Example Applies To

Returns a reference to a **Weighting** object that describes the size of a pie in relation to other pies in the same chart.

Syntax

object.**Weighting**

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

Visual Basic Reference

Visual Studio 6.0

WhatsThisButton Property

See Also Example Applies To

Returns or sets a value that determines whether the What's This button appears in the title bar of a **Form** object. Read-only at run time.

Syntax

object.**WhatsThisButton**

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

Settings

The settings for the **WhatsThisButton** property are:

Setting	Description
True	Turns display of the What's This Help button on.
False	(Default) Turns display of the What's This Help button off.

Remarks

The **WhatsThisHelp** property must be **True** for the **WhatsThisButton** property to be **True**. In addition, the following properties must also be set as shown:

- **ControlBox** property = **True**
- **BorderStyle** property = Fixed Single or Sizable
- **MinButton** and **MaxButton** = **False**
  Or
- **BorderStyle** property = Fixed Dialog

Visual Basic Reference

Visual Studio 6.0

WhatsThisHelp Property

See Also Example Applies To

Returns or sets a value that determines whether context-sensitive Help uses the What's This pop-up (provided by Help in 32-bit Windows operating systems) or the main Help window. Read-only at run time.

Syntax

object.**WhatsThisHelp** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A value that determines if Help uses the What's This pop-up, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The application uses one of the What's This access techniques to start Windows Help and load a topic identified by the **WhatsThisHelpID** property.
False	(Default) The application uses the F1 key to start Windows Help and load the topic identified by the **HelpContextID** property.

Remarks

There are three access techniques for providing What's This Help in an application. The **WhatsThisHelp** property must be set to **True** for any of these techniques to work.

- Providing a What's This button in the title bar of the form using the **WhatsThisButton** property. The mouse pointer changes into the What's This state (arrow with question mark). The topic displayed is identified by the **WhatsThisHelpID** property of the control clicked by the user.

- Invoking the **WhatsThisMode** method of a form. This produces the same behavior as clicking the What's This button without using a button. For example, you can invoke this method from a command on a menu in the menu bar of your application.

- Invoking the **ShowWhatsThis** method for a particular control. The topic displayed is identified by the **WhatsThisHelpID** property of the control.

Visual Basic Reference

Visual Studio 6.0

WhatsThisHelpID Property

See Also Example Applies To

Returns or sets an associated context number for an object. Use to provide context-sensitive Help for your application using the What's This pop-up in Help for 32-bit Windows operating systems.

Syntax

object.**WhatsThisHelpID** [= *number*]

The **WhatsThisHelpID** 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 specifying a Help context number, as described in Settings.

Settings

The settings for *number* are:

Setting	Description
0	(Default) No context number specified.
>0	An integer specifying the valid context number for the What's This topic associated with the object.

Remarks

Thirty-twobit Windows operating systems use the What's This button in the upper-right corner of the window to start Windows Help and load a topic identified by the **WhatsThisHelpID** property.

Visual Studio 6.0

Visual Basic: MSChart Control

Width Property (MSChart)

See Also Example Applies To

Returns or sets the width of a chart element, in points.

Syntax

object.**Width** [= *width*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
width	Single. The width of the chart element.

Visual Basic: Windows Controls

Visual Studio 6.0

Width Property (Panel Object)

See Also Example Applies To

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.

Visual Basic: Windows Controls

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
```

Visual Basic Extensibility Reference

Visual Studio 6.0

Width Property

See Also Example Applies To Specifics

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

Remarks

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

Visual Basic: MSChart Control

WidthToHeightRatio Property

See Also Example Applies To

Returns or sets the percentage of the chart height to be used as the chart width.

Syntax

object.**WidthToHeightRatio** [= *pctg*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
pctg	Single. The chart height percentage.

Visual Basic Extensibility Reference

Visual Studio 6.0

Window Property

See Also Example Applies To Specifics

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

Visual Basic Extensibility Reference

Window Property Example

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

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

Visual Basic Reference

Visual Studio 6.0

Windowless Property

See Also Example Applies To

Returns or sets a value that defines whether a UserControl is assigned a window handle (hWnd) at run time.

Syntax

object.**Windowless** [= *boolean*]

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

Part	Description
object	A UserControl object.
boolean	Determines whether a UserControl's **hWnd** property returns a valid hWnd.

Settings

The settings for Windowless are:

Setting	Description
True	The UserControl isnt assigned an **hWnd** at run time.
False	(Default) The UserControl is assigned an **hWnd** at run time.

Remarks

You can use the **Windowless** property to reduce the resource usage of your controls. By default, each instance of a UserControl object is assigned a window handle (referenced in the **hWnd** property) which consumes system resources. If the **Windowless** property is set to True, the **hWnd** property will always return 0.

In general, you should consider setting the **Windowless** property to True unless you need to access API calls which require an hWnd parameter or unless your control will act as a control container.

This property can only be set at design time.

Note Not all containers support Windowless controls. If a Windowless UserControl is sited on a container that doesnt support it, it will be assigned an hWnd and the **Windowless** property will be ignored.

Visual Basic Reference

Visual Studio 6.0

WindowList Property

See Also Example Applies To

Returns or sets a value that determines whether a **Menu** object maintains a list of the current MDI child windows in an **MDIForm** object. Read only at run time.

Syntax

object.**WindowList**

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

Settings

The **WindowList** property settings are:

Setting	Description
True	The **Menu** object maintains a list of open windows and displays a check mark next to the active window. Users can click a window name to activate that window.
False	(Default) The **Menu** doesn't maintain a list of open windows.

Remarks

Many multiple-document interface (MDI) applications, such as Microsoft Excel and Microsoft Word for Windows, have a Window menu containing a list of open MDI child windows. This property enables you to add this functionality to your application.

Only one **Menu** object on a form can have its **WindowList** property set to **True**.

When you select the WindowList check box in the Menu Editor for a **Menu** object, the list of open MDI child windows for the menu you're creating is displayed.

Visual Basic Reference

WindowList Property Example

This example creates some menu commands, illustrates the **WindowList** menu functionality, and shows how to enable your users to add new forms to a multiple-document interface (MDI) application. To try this example, create an **MDIForm** object with the Add MDI Form command on the Project menu. On Form1, set the **MDIChild** property to **True**, and create a menu named File. Select the WindowList box for the File menu. On your File menu, create a New command, set its **Name** property to FileMenu, and set its **Index** property to 0 to create a control array. Paste the code into the Declarations section of the form, and then press F5 to run the program. Choosing the New command on the File menu creates new MDI child forms. Their names are listed at the bottom of the File menu.

```
Private Sub Form_Load ()
   FileMenu(0).Caption = "&New"    ' Set access key in caption.
   Load FileMenu(1)   ' Create new menu item.
   FileMenu(1).Caption = "-"    ' Set separator.
   Load FileMenu(2)   ' Create new menu item.
   FileMenu(2).Caption = "E&xit"    ' Set caption and access key.
End Sub

Private Sub FileMenu_Click (Index As Integer)
   Select Case Index
      Case 0   ' Select New command.
         Dim NewForm As New Form1   ' Create a duplicate of Form1.
         ' Load NewForm and set a unique caption.
         NewForm.Caption = "Untitled" & Forms.Count
      Case 2   ' Select Exit command.
         End   ' End the program.
   End Select
End Sub
```

Visual Basic Extensibility Reference

Visual Studio 6.0

Windows Property

See Also Example Applies To Specifics

Returns the **Window** object, which represents a window in the Visual Basic IDE.

Syntax

object.**Window**

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

Visual Basic Reference

Visual Studio 6.0

WindowState Property

See Also Example Applies To

Returns or sets a value indicating the visual state of a form window at run time.

Syntax

object.**WindowState** [= *value*]

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

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Value	An integer specifying the state of the object, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
vbNormal	0	(Default) Normal.
vbMinimized	1	Minimized (minimized to an icon)
vbMaximized	2	Maximized (enlarged to maximum size)

Remarks

Before a form is displayed, the **WindowState** property is always set to Normal (0), regardless of its initial setting. This is reflected in the **Height**, **Left**, **ScaleHeight**, **ScaleWidth**, **Top**, and **Width** property settings. If a form is hidden after it's been shown, these properties reflect the previous state until the form is shown again, regardless of any changes made to the **WindowState** property in the meantime.

Visual Basic Reference

WindowState Property Example

This example hides a dialog box (Form2) when the parent form (Form1) is minimized and redisplays the dialog box when the parent form is returned to either an original or maximized state. To try this example, paste the code into the Declarations section of Form1 of an application that contains two forms. Press F5 to start the example. Move Form1 so you can see both forms, and then minimize or maximize the form and observe the behavior of Form2.

```
Private Sub Form_Load ()
   Form2.Show   ' Show Form2.
End Sub

Private Sub Form_Resize ()
   ' If parent form is minimized...
   If Form1.WindowState = vbMinimized Then
 ' ...hide Form2.
      Form2.Visible = False
   ' If parent form isn't minimized...
 Else
      ' ...restore Form2.
  Form2.Visible = True
   End If
End Sub
```

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

WordWrap Property (MSHFlexGrid)

SeeAlso Example Applies To

Returns or sets a value that determines whether a cell displays multiple lines of text or one long line of text.

Note Return characters, such as **Chr**(13), also force line breaks.

Syntax

object.**WordWrap** [=*Boolean*]

The **WordWrap** 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 text within a cell wraps.

Settings

The settings for *Boolean* are:

Part	Description
True	The cell text displays with multiple, wrapping lines of text.
False	The cell text displays as one long line of text. This is the default.

Remarks

The **MSHFlexGrid** displays text slightly faster when **WordWrap** is set to **False**.

Visual Basic: MSTab Control

Visual Studio 6.0

WordWrap Property (SSTab Control)

See Also Example Applies To

Returns or sets a value indicating whether the text on each tab is wrapped to the next line if it is too long to fit horizontally on the tab on an **SSTab** control.

Syntax

object.**WordWrap** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an **SSTab** control.
boolean	A Boolean expression that specifies whether the text on each tab will wrap to the next line if it does not fit horizontally, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The text wraps if it is too long to fit within the width of each tab.
False	(Default) The text doesn't wrap and will be truncated if it is too long.

Remarks

Use the **WordWrap** property to determine how an **SSTab** control displays the text on each tab. For example, a tabbed dialog that changes dynamically might have text that also changes. To make sure that text will not be truncated if it is too long, set the **WordWrap** property to **True,** the **TabMaxWidth** property to 0, and the **TabHeight** property to a height that allows you to view the longest piece of text.

Visual Studio 6.0

Visual Basic: MSChart Control

WordWrap Property(MSChart)

See Also Example Applies To

Returns or sets a value that determines whether text wraps.

Syntax

object.**WordWrap** [= *boolean*]

The **WordWrap** 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 whether text wraps, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	Text wraps.
False	(Default) Text does not wrap.

Visual Basic Reference

Visual Studio 6.0

WordWrap Property

See Also Example Applies To

Returns or sets a value indicating whether a **Label** control with its **AutoSize** property set to **True** expands vertically or horizontally to fit the text specified in its **Caption** property.

Syntax

object.**WordWrap** [= *boolean*]

The **WordWrap** 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 **Label** expands to fit the text, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The text wraps; the **Label** control expands or contracts vertically to fit the text and the size of the font. The horizontal size doesn't change.
False	(Default) The text doesn't wrap; the **Label** expands or contracts horizontally to fit the length of the text and vertically to fit the size of the font and the number of lines.

Remarks

Use this property to determine how a **Label** control displays its contents. For example, a graph that changes dynamically might have a **Label** containing text that also changes. To maintain a constant horizontal size for the **Label** and allow for increasing or decreasing text, set the **WordWrap** and **AutoSize** properties to **True**.

If you want a **Label** control to expand only horizontally, set **WordWrap** to **False**. If you don't want the **Label** to change size, set **AutoSize** to **False**.

Note If **AutoSize** is set to **False**, the text always wraps, regardless of the size of the **Label** control or the setting of the **WordWrap** property. This may obscure some text because the **Label** doesn't expand in any direction.

If both **AutoSize** and **WordWrap** are set to **True**, text will wrap without increasing the size of the **Label** control, unless a single word is entered that is larger than the width of the **Label**. In that case, the **AutoSize** property takes priority and the width of the **Label** increases to accommodate the long word.

Visual Basic Reference

WordWrap Property Example

This example puts text into two **Label** controls and uses the **WordWrap** property to illustrate their different behavior. To try this example, paste the code into the Declarations section of a form that contains two **Label** controls, and then press F5 and click the form to toggle the **WordWrap** property setting.

```
Private Sub Form_Load ()
   Dim Author1, Author2, Quote1, Quote2   ' Declare variables.
   Label1.AutoSize = True   ' Set AutoSize.
   Label2.AutoSize = True
   Label1.WordWrap = True   ' Set WordWrap.
   Quote1 = "I couldn't wait for success, so I went on without it."
   Author1 = "  - Jonathan Winters"
   Quote2 = "Logic is a system whereby one may go wrong with confidence."
   Author2 = "  - Charles Kettering"
   Label1.Caption = Quote1 & Chr(10) & Author1
   Label2.Caption = Quote2 & Chr(10) & Author2
End Sub

Private Sub Form_Click ()
   Label1.Width = 1440   ' Set width to 1 inch in twips.
   Label2.Width = 1440
   Label1.WordWrap = Not Label1.WordWrap ' Toggle WordWrap property.
   Label2.WordWrap = Not Label2.WordWrap
End Sub
```

Visual Basic: SysInfo Control

Visual Studio 6.0

WorkAreaHeight Property

See Also Example Applies To

Returns the height of the visible desktop adjusted for the Windows taskbar. Not available at design time.

Syntax

object.**WorkAreaHeight**

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

Remarks

When the taskbar appears along the top or bottom of the screen, the **WorkAreaHeight** property tells you the height of the visible desktop less the height of the taskbar.

Visual Basic: SysInfo Control

WorkAreaHeight Property Example

This example tests the size of the active form after a change in screen resolution and adjusts the size of the form if it exceeds the visible screen area. To run this example, put a **SysInfo** control on a form. Paste this code into the DisplayChanged event of the **SysInfo** control. Run the example and change the screen resolution.

```
Private Sub SysInfo1_DisplayChanged()
    If Screen.ActiveForm.Width > SysInfo1.WorkAreaWidth Then
        Screen.ActiveForm.Left = SysInfo1.WorkAreaLeft
        Screen.ActiveForm.Width = SysInfo1.WorkAreaWidth
    End If
    If Screen.ActiveForm.Height > SysInfo1.WorkAreaHeight Then
        Screen.ActiveForm.Top = SysInfo1.WorkAreaTop
        Screen.ActiveForm.Height = SysInfo1.WorkAreaHeight
    End If
End Sub
```

Visual Basic: SysInfo Control

Visual Studio 6.0

WorkAreaLeft Property

See Also Example Applies To

Returns the coordinate for the left edge of the visible desktop adjusted for the Windows taskbar. Not available at design time.

Syntax

object.**WorkAreaLeft**

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

Visual Basic: SysInfo Control

WorkAreaLeft Property Example

This example tests the size of the active form after a change in screen resolution and adjusts the size of the form if it exceeds the visible screen area. To run this example, put a **SysInfo** control on a form. Paste this code into the DisplayChanged event of the **SysInfo** control. Run the example then change the screen resolution.

```
Private Sub SysInfo1_DisplayChanged()
    If Screen.ActiveForm.Width > SysInfo1.WorkAreaWidth Then
        Screen.ActiveForm.Left = SysInfo1.WorkAreaLeft
        Screen.ActiveForm.Width = SysInfo1.WorkAreaWidth
    End If
    If Screen.ActiveForm.Height > SysInfo1.WorkAreaHeight Then
        Screen.ActiveForm.Top = SysInfo1.WorkAreaTop
        Screen.ActiveForm.Height = SysInfo1.WorkAreaHeight
    End If
End Sub
```

Visual Basic: SysInfo Control

Visual Studio 6.0

WorkAreaTop Property

See Also Example Applies To

Returns the coordinate for the top edge of the visible desktop adjusted for the Windows taskbar. Not available at design time.

Syntax

object.**WorkAreaTop**

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

Visual Basic: SysInfo Control

WorkAreaTop Property Example

This example tests the size of the active form after a change in screen resolution and adjusts the size of the form if it exceeds the visible screen area. To run this example, put a **SysInfo** control on a form. Paste this code into the DisplayChanged event of the **SysInfo** control. Run the example, then change the screen resolution.

```
Private Sub SysInfo1_DisplayChanged()
    If Screen.ActiveForm.Width > SysInfo1.WorkAreaWidth Then
        Screen.ActiveForm.Left = SysInfo1.WorkAreaLeft
        Screen.ActiveForm.Width = SysInfo1.WorkAreaWidth
    End If
    If Screen.ActiveForm.Height > SysInfo1.WorkAreaHeight Then
        Screen.ActiveForm.Top = SysInfo1.WorkAreaTop
        Screen.ActiveForm.Height = SysInfo1.WorkAreaHeight
    End If
End Sub
```

Visual Basic: SysInfo Control

Visual Studio 6.0

WorkAreaWidth Property

See Also Example Applies To

Returns the width of the visible desktop adjusted for the Windows taskbar. Not available at design time.

Syntax

object.**WorkAreaWidth**

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

Remarks

When the taskbar appears along the left or right edge of the screen, the **WorkAreaWidth** property tells you the width of the visible desktop minus the width of the taskbar.

Visual Basic: SysInfo Control

WorkAreaWidth Property Example

This example tests the size of the active form after a change in screen resolution and adjusts the size of the form if it exceeds the visible screen area. To run this example, put a **SysInfo** control on a form. Paste this code into the DisplayChanged event of the **SysInfo** control. Run the example, then change the screen resolution.

```
Private Sub SysInfo1_DisplayChanged()
    If Screen.ActiveForm.Width > SysInfo1.WorkAreaWidth Then
        Screen.ActiveForm.Left = SysInfo1.WorkAreaLeft
        Screen.ActiveForm.Width = SysInfo1.WorkAreaWidth
    End If
    If Screen.ActiveForm.Height > SysInfo1.WorkAreaHeight Then
        Screen.ActiveForm.Top = SysInfo1.WorkAreaTop
        Screen.ActiveForm.Height = SysInfo1.WorkAreaHeight
    End If
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

Wrap Property

See Also Example Applies To

Sets or returns a value that determines whether the control's **Value** property wraps around to the beginning or end once it reaches the **Max** or **Min** value.

Syntax

object.**Wrap** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A boolean expression that determines whether the **UpDown** control wraps its Value property, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	The **UpDown** control wraps the **Value** property to the beginning or end when the user moves past the **Max** or **Min** properties using the arrow buttons.
False	(Default) The **UpDown** control doesn't wrap the **Value** property.

Remarks

When **Wrap** is **True**, the **Value** property wraps according to the value of the **Min**, **Max**, and **Increment** properties.

When the **Value** property wraps, its first jump is to the **Min** value. For example, if **Min** is 20, **Max** is 70, **Increment** is 10, **Value** is 70, and the user clicks the up or right arrow button, the **Value** property wraps to 20.

Visual Basic: Windows Controls

Wrap Property Example

This example uses the **Wrap** property with the **UpDown** control to create a wrapping spinner control with values that range from 10 to 70 and an increment of 10. To try this example, place a **TextBox** control and an **UpDown** control on your form and add the following code:

```
Private Sub Form_Load()
   UpDown1.BuddyControl = Text1
   With UpDown1
      .Min = 10
      .Max = 70
      .Increment = 10
      .Wrap = True
      .SyncBuddy = True
   End With

   ' So the TextBox reflects the starting value
   Text1.Text = UpDown1.Value
End Sub
```

Visual Basic: DataGrid Control

Visual Studio 6.0

WrapCellPointer Property

See Also Example Applies To

Sets or returns a value that determines the behavior of the arrow keys.

Syntax

object.**WrapCellPointer** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines the behavior of arrow keys, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	The cell pointer will wrap from the last column to the first in the next row (or from the first column to the last in the previous row).
False	(Default) The cell pointer will not wrap to the next (or previous) row, but will stop at the last (or first) column of the current row.

Remarks

If **TabAcrossSplits** is **False**, the cell pointer will wrap only within the current split. If **TabAcrossSplits** is **True**, the cell pointer will move from one split to the next before wrapping occurs.

If **TabAction** is set to 2 - Grid Navigation, the tab key will behave like the arrow keys, and will automatically wrap to the next or previous cell.

Visual Basic: Windows Controls

Visual Studio 6.0

Wrappable Property

See Also Example Applies To

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.

Visual Basic: DataGrid Control

Visual Studio 6.0

WrapText Property (Column Object)

See Also Example Applies To

Sets or returns a value indicating whether an object word wraps text at cell boundaries.

Syntax

object.**WrapText** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines whether an object word wraps, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	A line break occurs before words that would otherwise be partially displayed.
False	(Default) No line break occurs and text is clipped at the cell's right edge.

Remarks

Use this property in conjunction with the **RowHeight** property to produce multi-line displays.