Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

BackColor, BackColorBkg, BackColorFixed, BackColorSel Properties

See Also Example Applies To

Returns or sets the background color of various elements of the **MSHFlexGrid**.

Syntax

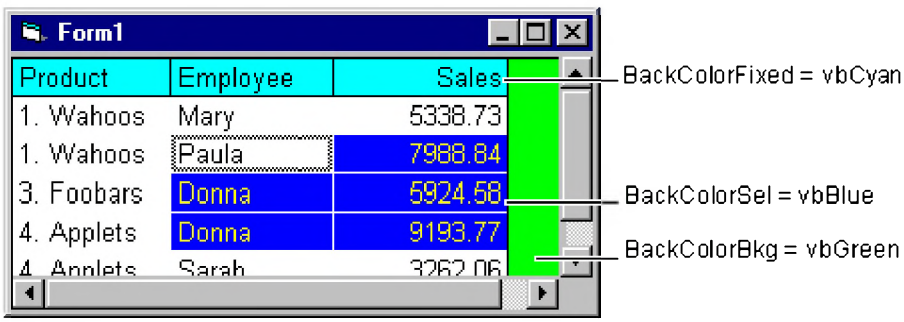
object.**BackColor** [=*color*]
object.**BackColorBkg** [=*color*]
object.**BackColorFixed** [=*color*]
object.**BackColorSel** [=*color*]

Syntax for the **BackColor**, **BackColorBkg**, **BackColorFixed**, and **BackColorSel** properties has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
color	A numeric expression that specifies the color.

Remarks

The picture below shows the part of the **MSHFlexGrid** to which the properties refer:



BackColor affects the color of all non-fixed cells. To set the background color of individual cells, use the **CellBackColor** property.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

BackColor, BackColorBkg, BackColorFixed, BackColorSel Properties Example

The following example uses the **BackColorBkg**, **BackColorFixed**, and **BackColorSel** properties in the **MSHFlexGrid**. It resets the colors of the control's background, selected background, and fixed-cell background randomly twice each second for the **MSHFlexGrid**. To use this example, paste the code into the Declarations section of a form with a **Timer** control and an **MSHFlexGrid** with the names Timer1 and MSHFlexGrid1 respectively, then load the form.

Note If you are using the **MSFlexGrid**, substitute "MSHFlexGrid1" with "MSFlexGrid1."

```
Private Sub Form_Load ()
   Timer1.Interval =500
End Sub

Private Sub Timer1_Timer ()
   MSHFlexGrid1.BackColorBkg =QBColor(Rnd * 15)
   MSHFlexGrid1.BackColorFixed =QBColor(Rnd * 10)
   MSHFlexGrid1.BackColorSel =QBColor(Rnd * 10)
End Sub
```

Visual Basic Reference

Visual Studio 6.0

BackColor, ForeColor Properties

See Also Example Applies To

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

Syntax

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

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

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

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

Settings

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

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

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

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

Remarks

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

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

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

To display text in the Windows operating environment, both the text and background colors must be solid. If the text or background colors you've selected aren't displayed, one of the selected colors may be dithered that is, comprised of up to three different-colored pixels. If you choose a dithered color for either the text or background, the nearest solid color will be substituted.

Visual Basic Reference

BackColor, ForeColor Properties Example

This example resets foreground and background colors randomly twice each second for a form and **PictureBox** control. To try this example, paste the code into the Declarations section of a form that contains a **PictureBox** control and a **Timer** control, and then press F5.

```
Private Sub Form_Load ()
   Timer1.Interval = 500
End Sub

Private Sub Timer1_Timer ()
   BackColor = QBColor(Rnd * 15)
   ForeColor = QBColor(Rnd * 10)
   Picture1.BackColor = QBColor(Rnd * 15)
   Picture1.ForeColor = QBColor(Rnd * 10)
End Sub
```

Visual Basic Reference

Visual Studio 6.0

BackColor, ForeColor Properties (ActiveX Controls)

See Also Example Applies To

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

Syntax

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

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

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

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

Settings

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

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

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

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

Remarks

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

To display text in the Windows operating environment, both the text and background colors must be solid. If the text or background colors you've selected aren't displayed, one of the selected colors may be dithered that is, comprised of up to three different-colored pixels. If you choose a dithered color for either the text or background, the nearest solid color will be substituted.

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

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

BackColorBand, BackColorHeader, BackColorIndent, BackColorUnpopulated Properties

See Also Example Applies To

- **BackColorBand** Returns or sets the background color of the band area of the **MSHFlexGrid**.
- **BackColorHeader** Returns or sets the background color of the header area of the **MSHFlexGrid**.
- **BackColorIndent** Returns or sets the background color of the indented area of the **MSHFlexGrid**.
- **BackColorUnpopulated** Returns or sets the background color of the unpopulated area of the **MSHFlexGrid**.

Syntax

object.**BackColorBand (***BandNumber***)** [=*color*]
object.**BackColorHeader (***BandNumber***)** [=*color*]
object.**BackColorIndent (***BandNumber***)** [=*color*]
object.**BackColorUnpopulated** [=*color*]

Syntax for the **BackColorBand**, **BackColorHeader**, **BackColorIndent**, and **BackColorUnpopulated** properties has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
BandNumber	Required. A Long value that specifies the band to be affected.
color	A numeric expression that specifies the color.

Visual Studio 6.0

Visual Basic: MSChart Control

Backdrop Property

See Also Example Applies To

Returns a reference to a **Backdrop** object that describes the shadow, pattern, or picture behind a chart or chart element.

Syntax

object.**Backdrop**

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

Visual Basic Reference

Visual Studio 6.0

BackStyle Property

See Also Example Applies To

Returns or sets a value indicating whether a **Label** control or the background of a **Shape** control is transparent or opaque.

Syntax

object.**BackStyle** [= *number*]

The **BackStyle** 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 transparency, as described in Settings.

Settings

The settings for *number* are:

Setting	Description
0	Transparent background color and any graphics are visible behind the control.
1	(Default) Opaque the control's **BackColor** property setting fills the control and obscures any color or graphics behind it.

Remarks

You can use the **BackStyle** property to create transparent controls when you're using a background color on a **Form** object or **PictureBox** control or when you want to place a control over a graphic. Use an opaque control when you want it to stand out.

A control's **BackColor** property is ignored if **BackStyle** = 0.

Visual Basic: Windows Controls

Visual Studio 6.0

BackStyle Property (Animation Control)

See Also Example Applies To

Returns or sets a value that determines whether the **Animation** control draws the animation on a transparent background or on the background color specified in the animation clip. Read-only at run time.

Syntax

object.**BackStyle** [= *value*]

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

Part	Description
object	An object expression that evaluates to an **Animation** control.
value	A numeric expression specifying transparency, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
0	(Default) Transparentbackground color of the control is visible.
1	Opaque the background color specified in the animation clip fills the control and obscures any color behind it.

Remarks

You can use the **BackStyle** property to run an animation that shows the background color of the **Animation** control rather than the background color of the animation itself.

Data Type

Integer (Boolean)

Visual Basic Reference

Visual Studio 6.0

BackStyle Property (UserControl Object)

See Also Example Applies To

Returns or sets a value indicating the type of the controls background.

Syntax

object.**BackStyle** [= *enum*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
enum	An enumerated value that determines how the background of the control will be displayed, as described in Settings.

Settings

The settings for *enum* are:

Setting	Description
1- Opaque	(Default) Opaque background. All mouse events are received by the control.
2- Invisible	Applies only when the **Windowless** property is set to **True**. Otherwise the behavior is the same as with a Transparent **BackStyle**.

Remarks

When *enum* is set to 2, the appearance and behavior of controls with the **Windowless** property set to **True** are based on the settings of the **MaskPicture**, **MaskColor**, **HitBehavior**, and **ClipBehavior** properties.

Visual Basic: Windows Controls

Visual Studio 6.0

BandBorders Property

See Also Example Applies To

Returns or sets a value indicating whether a **CoolBar** control displays narrow lines to separate the bands.

Syntax

object.**BandBorders** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to a **CoolBar** control.
boolean	A Boolean expression specifying whether the band border is visible or hidden.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Borders are visible.
False	Borders are hidden.

Remarks

To remove the borders between bands, set the **BandBorders** property to **False** at design time. Setting this property in code enables you to hide or display borders between bands at run time in response to a particular event. Setting this property affects all bands within a **CoolBar** control.

Note The BandBorders property has no effect on the outside border of the **CoolBar** itself. The **CoolBar** border is always displayed.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

BandDisplay Property (MSHFlexGrid)

SeeAlso Example Applies To

Specifies how the bands display within the **MSHFlexGrid**.

Syntax

object.**DisplayBandSettings** [=*value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A numeric expression that specifies how the bands display within the **MSHFlexGrid**, as described in Settings.

Settings

The settings for *value* are:

Setting	Value	Description
flexBandDisplayHorizontal	0	Default. The bands within the **MSHFlexGrid** display horizontally (across).
flexBandDisplayVertical	1	The bands within the **MSHFlexGrid** display vertically (up and down).

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

BandExpandable Property (MSHFlexGrid)

SeeAlso Example Applies To

Returns and sets a value that determines whether the row in the current band can be expanded and collapsed. The current band is defined by the **Col** and **Row** properties. This is a read-only property at run time and is not available at design time.

Syntax

object.**BandExpandable(***number***)** [=*Boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A Long value indicating the number of the band in the **MSHFlexGrid**.
Boolean	A Boolean expression that determines whether a specific band can be expanded and collapsed.

Settings

The settings for *Boolean* are:

Setting	Description
True	Default. The band specified can be expanded and collapsed.
False	The band specified cannot be expanded and collapsed.

Remarks

When a band is expandable, a standard plus (+) and minus () bitmap displays on the left of the first column within the band. This bitmap can be overridden by setting the **ExpandPicture** and **CollapsePicture** properties. When the grid first displays, all of the bands are collapsible by default.

Setting this property disables the user's capacity to expand or collapse the band; it doesnt cause the rows within the specified band to expand or collapse. For example, if the child rows are shown before setting this property to **False**, the child rows continue to be shown after the setting.

For a band to be expandable, it must have at least one sub-band. Without a sub-band, the **BandExpandable** property is ignored.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

BandIndent Property (MSHFlexGrid)

SeeAlso Example Applies To

Specifies the number of columns by which to indent a band.

Syntax

object.**BandIndent (*BandIndex*)** [=*number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
BandNumber	Required. A Long value that specifies the band to be affected.
number	A Long value that specifies the band to be indented in the **MSHFlexGrid**. The default setting is 0.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

BandLevel Property (MSHFlexGrid)

SeeAlso Example Applies To

Returns the band number that contains the current cell. The band numbers begin at 0. The current cell is defined by the **Col** and **Row** properties. This property is not available at design time.

Syntax

object.**BandLevel** [=*number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	An integer or constant that specifies the band number that contains the current cell.

Visual Basic: Windows Controls

Visual Studio 6.0

Bands Property

See Also Example Applies To

Returns a reference to a **CoolBar** controls collection of **Band** objects.

Syntax

object.**Bands**

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

Remarks

You can manipulate **Band** objects using standard collection methods (for example, the Add and Remove methods). Each element in the collection can be accessed by its index, the value of the **Index** property, or by a unique key, the value of the **Key** property.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

Bands Property (MSHFlexGrid)

See Also Example Applies To

Returns the total number of bands in an **MSHFlexGrid**. The **MSHFlexGrid** always has a minimum of one band. When the **MSHFlexGrid** is bound to a standard Recordset, the entire **MSHFlexGrid** is treated as one band.

Syntax

object.**Bands** [=*value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	The total number of bands in an **MSHFlexGrid**.

Remarks

This property is read-only. The number of Recordsets in the hierarchy of Recordsets with which the control is bound defines the value.

See the **Col** and **Row** properties.

Visual Studio 6.0

Visual Basic: MSChart Control

BarGap Property

See Also Example Applies To

Returns or sets the spacing of two-dimensional bars or clustered three-dimensional bars within a category.

Syntax

object.**BarGap** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	Single. The bar spacing value. This is measured as a percentage of the bar width. A value of 0 results in the bars touching. A value of 100 means the gap between the bars is as wide as the bars.

Visual Studio 6.0

Visual Basic: MSChart Control

BaseHeight Property

See Also Example Applies To

Returns or sets the height of the three-dimensional chart base in points.

Syntax

object.**BaseHeight** [= *height*]

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

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

Visual Basic: Page Designer

Visual Studio 6.0

BaseWindow Property

See Also Example Applies To

Returns the topmost window object from the DHTML object model. The topmost window represents the browser window and any frames it contains.

Syntax

object.**BaseWindow**

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

Remarks

The objects available from the topmost window (for example, history, event, navigator, and so on) are always available from the **BaseWindow** property. **BaseWindow** in Visual Basic corresponds to the Window object in Internet Explorer 4.0 browsers Dynamic HTML object model.

Visual Basic: MSChart Control

Basis Property

See Also Example Applies To

Returns or sets the type of weighting used to determine pie size on a chart.

Syntax

object.**Basis** [= *type*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
type	A **VtChPieWeightBasis** constant that identifies the weighting type.

Visual Basic: RDO Data Control

Visual Studio 6.0

BatchCollisionCount Property

See Also Example Applies To

Returns a value that specifies the number of rows that did not complete the last batch-mode update.

Syntax

object.**BatchCollisionCount**

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

Return Values

The **BatchCollisionCount** property return value is a Long expression that specifies the number of failing rows or 0 if all rows were processed.

Remarks

This property indicates how many rows encountered collisions or otherwise failed to update during the last batch update attempt. The value of this property corresponds to the number of bookmarks in the **BatchCollisionRows** array.

By setting the working **rdoResultset** object's **Bookmark** property to bookmark values in the **BatchCollisionRows** array, you can position to each row that failed to complete the most recent **BatchUpdate** operation.

After the collision rows are corrected, the **BatchUpdate** method can be called again. At this point RDO attempts another batch update, and the **BatchCollisionRows** property again reflects the set of rows that failed the second attempt. Any rows that succeeded in the previous attempt are not sent in the current attempt, as they now have a **Status** of **rdRowUnmodified**. This process can continue as long as collisions occur, or until you abandon the updates and close the result set.

Visual Basic: RDO Data Control

Visual Studio 6.0

BatchCollisionRows Property

See Also Example Applies To

Returns an array of bookmarks indicating the rows that generated collisions in the last batch update operation.

Syntax

object.**BatchCollisionRows**

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

Return Values

The **BatchCollisionRows** property return value is a **Variant**(string) expression as described in Remarks.

Remarks

This property contains an array of bookmarks to rows that encountered a collision during the last invocation of the **BatchUpdate** method. The number of elements in the array is indicated by the **BatchCollisionCount** property.

By setting the working **rdoResultset** object's **Bookmark** property to bookmark values in the **BatchCollisionRows** array, you can position to each row that failed to complete the most recent **BatchUpdate** operation.

After the collision rows are corrected, the **BatchUpdate** method can be called again. At this point RDO attempts another batch update, and the **BatchCollisionRows** property again reflects the set of rows that failed the second attempt. Any rows that succeeded in the previous attempt are not sent in the current attempt, as they now have a **Status** of **rdRowUnmodified**. This process can continue as long as collisions occur, or until you abandon the updates and close the result set.

This array is re-created each time the **BatchUpdate** method executes.

Visual Basic: RDO Data Control

Visual Studio 6.0

BatchConflictValue Property

See Also Example Applies To

Returns a value currently in the database that is newer than the **Value** property as determined by an optimistic batch update conflict.

Syntax

object.**BatchConflictValue**

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

Return Values

The **BatchConflictValue** property return value is a **Variant**(String) expression as described in Remarks.

Remarks

This property contains the value of the column that is currently in the database on the server. During an optimistic batch update, a collision may occur where a second client modified the same column and row in between the time the first client fetched the data and the update attempt. When this happens, the value that the second client set will be accessible through this property.

Visual Basic: RDO Data Control

Visual Studio 6.0

BatchSize Property

See Also Example Applies To

Returns or sets a value that specifies the number of statements sent back to the server in each batch.

Syntax

object.**BatchSize** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **Long** integer representing the number of statements sent back to the server in each batch. The default value is 15.

Remarks

This property determines the batch size used when sending statements to the server during an optimistic batch update. The value of the property determines the number of statements sent to the server in one command buffer. By default, 15 statements are sent to the server in each batch. This property can be changed at any time. If a DBMS doesnt support statement batching, you can set this property to 1, causing each statement to be sent separately.

Visual Basic: SysInfo Control

Visual Studio 6.0

BatteryFullTime Property

See Also Example Applies To

Returns a value that indicates the full charge life of the battery.

Syntax

object.**BatteryFullTime**

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

Remarks

The **BatteryFullTime** property returns the number of seconds of life in the battery when at full charge. If the batterys full charge life time is unknown, this property returns the value &HFFFFFFFF.

Visual Basic: SysInfo Control

BatteryFullTime Property Example

This example displays the amount of total battery time in hours and minutes. To run this example, put a **SysInfo** control and a **CommandButton** control on a form. Paste this code into the Click event of the **CommandButton** control, then run the example.

```
Private Sub Command1_Click()
    ' BatteryLifeTime Property Example
    If SysInfo1.BatteryLifeTime <> &HFFFFFFFF Then
        Dim TimeLeft As String
        Dim TimeTotal As String
        Dim temp
        temp = TimeSerial(0, 0, SysInfo1.BatteryLifeTime)
        TimeLeft = Format(temp, "h:mm")
        temp = TimeSerial(0, 0, SysInfo1.BatteryFullTime)
        TimeTotal = Format(temp, "h:mm")
        MsgBox TimeLeft & " time left from " & TimeTotal & " total."
    Else
        MsgBox "Cannot determine remaining battery time."
    End If
End Sub
```

Visual Basic: SysInfo Control

Visual Studio 6.0

BatteryLifePercent Property

See Also Example Applies To

Returns the percentage of full battery power remaining.

Syntax

object.**BatteryLifePercent**

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

Settings

The **BatteryLifePercent** property settings are:

Setting	Description
0 - 100	Percentage of full battery charge remaining.
255	Battery charge status is unknown.

Visual Basic: SysInfo Control

BatteryLifePercent Property Example

This example uses a **ProgressBar** control on a form to show the status of the system battery's remaining power. To run this example, put a **SysInfo** control, a **ProgressBar** control and a **Timer** control on a form. Paste this code into the Timer event of the **Timer** control. Set the **Interval** property of the **Timer** control to 5000, then run the example.

```
Private Sub Timer1_Timer()
   If SysInfo1.BatteryLifePercent <> 255 Then
      ProgressBar1.Value = SysInfo1.BatteryLifePercent
      ProgressBar1.Enabled = True
   Else
      ProgressBar1.Value = 0
      ProgressBar1.Enabled = False
   End If
End Sub
```

Visual Basic: SysInfo Control

Visual Studio 6.0

BatteryLifeTime Property

See Also Example Applies To

Returns a value that indicates the remaining life of the battery.

Syntax

object.**BatteryLifeTime**

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

Remarks

The **BatteryLifeTime** property returns the number of seconds of life remaining in the battery. If the battery life time is unknown, this property returns the value &HFFFFFFFF.

Visual Basic: SysInfo Control

BatteryLifeTime Property Example

This example displays the amount of remaining battery time in hours and minutes. To run this example, put a **SysInfo** control and a **CommandButton** control on a form. Paste this code into the Click event of the **CommandButton** control. Then run the example.

```
Private Sub Command1_Click()
    ' BatteryLifeTime Property Example
    If SysInfo1.BatteryLifeTime <> &HFFFFFFFF Then
        Dim TimeLeft As String
        Dim TimeTotal As String
        Dim temp
        temp = TimeSerial(0, 0, SysInfo1.BatteryLifeTime)
        TimeLeft = Format(temp, "h:mm")
        temp = TimeSerial(0, 0, SysInfo1.BatteryFullTime)
        TimeTotal = Format(temp, "h:mm")
        MsgBox TimeLeft & " time left from " & TimeTotal & " total."
    Else
        MsgBox "Cannot determine remaining battery time."
    End If
End Sub
```

Visual Basic: SysInfo Control

Visual Studio 6.0

BatteryStatus Property

See Also Example Applies To

Returns a value that indicates the status of the batterys charge.

Syntax

object.**BatteryStatus**

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

Settings

The **BatteryStatus** property settings are:

Setting	Description
1	Battery charge is high.
2	Battery charge is low.
4	Battery charge is critical.
8	Battery is charging.
128	The system has no battery.
255	Battery charge status is unknown.

Visual Basic: SysInfo Control

BatteryStatus Property Example

This example updates a **Label** control each time there is a change in power status so the application displays current battery status information. To run this example, put a **SysInfo** control and a **Label** control on a form. Paste this code into the PowerStatusChanged event of the **SysInfo** control. Then run the example.

```
Private Sub SysInfo1_PowerStatusChanged()
   Select Case SysInfo1.BatteryStatus
      Case 1
         Label1.Caption = "Battery OK"
      Case 2
         Label1.Caption = "Battery Low"
      Case 4
         Label1.Caption = "Battery Critical"
      Case 8
         Label1.Caption = "Battery Charging"
      Case 128, 255
         Label1.Caption = "No Battery Status"
   End Select
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

Bevel Property (Panel Object)

See Also Example Applies To

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

Syntax

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

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

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

Settings

The settings for *value* are:

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

Visual Basic: Windows Controls

Bevel Property (Panel Object) Example

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

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

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

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

Visual Basic Reference

Visual Studio 6.0

Bindable Property

See Also Example Applies To

Returns or sets the **Bindable** property associated with a **Member** object.

Syntax

object.**Bindable**

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

Visual Basic: RDO Data Control

Visual Studio 6.0

BindThreshold Property

See Also Example Applies To

Returns or sets a value specifying the largest column that will be automatically bound under ODBC.

Syntax

object.**BindThreshold** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Long expression as described in Remarks.

Remarks

The default value for **BindThreshold** is 1024 bytes.

Several data types support sizes that are far too large to handle using conventional string or byte array techniques. For these columns, you should use the **GetChunk** and **AppendChunk** methods. However, use of these methods is not required you can simply address the **Value** property assuming the size of the chunk data does not exhaust your resources.

By setting the **BindThreshold** property, you can set the maximum size of chunk that RDO automatically binds to strings. Columns larger than the **BindThreshold** value require use of the **GetChunk** method to retrieve data. The **ChunkRequired** property indicates if the column requires use of **AppendChunk** and **GetChunk** methods by comparing the column's data size against the **BindThreshold** value.

Visual Studio 6.0

Visual Basic: MSChart Control

Blue Property

See Also Example Applies To

Returns or sets the blue component of the RGB value in a chart.

Syntax

object.**Blue** [=*b*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
b	Integer. The blue value.

Remarks

RGB specifies the relative intensity of red, green, and blue to cause a specific color to be displayed. The valid range for a normal RGB color is 0 to 16,777,215. The value for any argument to RGB that exceeds 255 is assumed to be 255.

Visual Basic: RDO Data Control

Visual Studio 6.0

BOF, EOF Properties (Remote Data)

See Also Example Applies To

- **BOF** returns a value that indicates whether the current row position is before the first row in an **rdoResultset**.
- **EOF** returns a value that indicates whether the current row position is after the last row in an **rdoResultset**.

Syntax

object.**BOF**

object.**EOF**

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

Return Values

The **BOF** property return values are:

Value	Description
True	The current row position is before the first row.
False	The current row position is on or after the first row.

The **EOF** property return values are:

Value	Description
True	The current row position is after the last row.
False	The current row position is on or before the last row.

Remarks

The **BOF** and **EOF** return values are determined by the location of the current row pointer if this pointer is valid. If either **BOF** or **EOF** is **True**, there is no current row, and any attempt to reference **rdoResultset** data results in a trappable error.

You can use the **BOF** and **EOF** properties to determine whether an **rdoResultset** object contains rows or whether you've gone beyond the limits of an **rdoResultset** as you move from row to row.

If you open an **rdoResultset** containing no rows, **BOF** and **EOF** are set to **True**, and the result set's **RowCount** property setting is 0. When you open an **rdoResultset** that contains at least one row, the first row is the current row and **BOF** and **EOF** are **False**; they remain **False** until you move beyond the beginning or end of the **rdoResultset** using the **MovePrevious** or **MoveNext** method, respectively. When you move beyond the beginning or end of the **rdoResultset**, there is no current row.

If you delete the last remaining row in the **rdoResultset** object, **BOF** and **EOF** might remain **False** until you attempt to reposition the current row.

If you use the **MoveLast** method on an **rdoResultset** containing rows, the last row becomes the current row; if you then use the **MoveNext** method, the current row becomes invalid and **EOF** is set to **True**. Conversely, if you use the **MoveFirst** method on an **rdoResultset** containing rows, the first row becomes the current row; if you then use the **MovePrevious** method, there is no current row and **BOF** is set to **True**.

Typically, when you work with all the rows in an **rdoResultset**, your code will loop through the rows using **MoveNext** until the **EOF** property is set to **True**.

If you use **MoveNext** while **EOF** is set to **True** or **MovePrevious** while **BOF** is set to **True**, a trappable error occurs.

This table shows which *Move* methods are allowed with different combinations of **BOF** and **EOF**.

	MoveFirst, MoveLast	**MovePrevious, Move < 0**	**Move 0**	**MoveNext, Move > 0**
BOF = True, EOF = False	Allowed	Error	Error	Allowed
BOF = False, EOF = True	Allowed	Allowed	Error	Error
Both **True**	Error	Error	Error	Error
Both **False**	Allowed	Allowed	Allowed	Allowed

Allowing a *Move* method doesn't mean that the method will successfully locate a row. It merely indicates that an attempt to perform the specified *Move* method is allowed and won't generate an error. The state of the **BOF** and **EOF** properties may change as a result of the attempted Move.

Effect of specific methods on **BOF** and **EOF** settings:

- An **OpenResultset** method internally invokes a **MoveFirst**. Therefore, an **OpenResultset** on an empty set of rows results in **BOF** and **EOF** being set to **True**.
- All *Move* methods that successfully locate a row set both **BOF** and **EOF** to **False**.
- For dynamic-type **rdoResultset** objects, any **Delete** method, even if it removes the only remaining row from an **rdoResultset**, won't change the setting of **BOF** or **EOF**.
- For other types of **rdoResultset** objects, the **BOF** and **EOF** properties are unchanged as add and delete operations are made because result set membership is fixed.

Visual Basic Reference

Visual Studio 6.0

BOFAction, EOFAction Properties

See Also Example Applies To

Returns or sets a value indicating what action the **Data** control takes when the **BOF** or **EOF** properties are **True**.

Syntax

object.**BOFAction** [= *integer*]

object.**EOFAction** [= *integer*]

The **BOFAction** and **EOFAction** property syntax's have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list
integer	An integer value that specifies an action, as described in Settings

Settings

For the **BOFAction** property, the settings for *integer* are:

Setting	Value	Description
vbBOFActionMoveFirst	0	**MoveFirst** (Default): Keeps the first record as the current record.
vbBOFActionBOF	1	**BOF**: Moving past the beginning of a **Recordset** triggers the **Data** control Validate event on the first record, followed by a Reposition event on the invalid (**BOF**) record. At this point, the Move Previous button on the **Data** control is disabled.

For the **EOFAction** property, the settings for *integer* are:

Setting	Value	Description
vbEOFActionMoveLast	0	**MoveLast** (Default): Keeps the last record as the current record.

vbEOFActionEOF	1	**EOF**: Moving past the end of a **Recordset** triggers the **Data** control's Validation event on the last record, followed by a Reposition event on the invalid (**EOF**) record. At this point, the MoveNext button on the **Data** control is disabled.
vbEOFActionAddNew	2	**AddNew**: Moving past the last record triggers the **Data** control's Validation event to occur on the current record, followed by an automatic **AddNew**, followed by a Reposition event on the new record.

Remarks

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

If you set the **EOFAction** property to **vbEOFActionAddNew**, once the user moves the current record pointer to **EOF** using the **Data** control, the current record is positioned to a new record in the copy buffer. At this point you can edit the newly added record. If you make changes to the new record and the user subsequently moves the current record pointer using the **Data** control, the record is automatically appended to the **Recordset**. If you don't make changes to this new record, and reposition the current record to another record, the new record is discarded. If you use the **Data** control to position to another record while positioned over this new record, another new record is created.

When you use code to manipulate **Recordsets** created with the **Data** control, the **EOFAction** property has no effect it only takes effect when manipulating the **Data** control with the mouse.

In situations where the **Data** control **Recordset** is returned with no records, or after the last record has been deleted, using the **vbEOFActionAddNew** option for the **EOFAction** property greatly simplifies your code because a new record is always editable as the current record. If this option is not enabled, you are likely to trigger a "No current record" error.

Data Type

Integer

Visual Basic: RDO Data Control

Visual Studio 6.0

BOFAction, EOFAction Properties (Remote Data)

See Also Example Applies To

Returns or sets a value indicating what action the **RemoteData control** takes when the **BOF** or **EOF** property is **True**.

Syntax

object.**BOFAction** [= *value*]

object.**EOFAction** [= *value*]

The **BOFAction** and **EOFAction** property syntaxes have these parts:

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

Settings

For the **BOFAction** property, the settings for *value* are:

Constant	Value	Description
rdMoveFirst	0	**MoveFirst** (Default): Keeps the first row as the current row.
rdBOF	1	**BOF**: Moving past the beginning of an **rdoResultset** triggers the **RemoteData** control's Validate event on the first row, followed by a Reposition event on the invalid (**BOF**) row. At this point, the Move Previous button on the **RemoteData** control is disabled.

For the **EOFAction** property, the settings for *value* are:

Constant	Value	Description

rdMoveLast	0	**MoveLast** (Default): Keeps the last row as the current row.
rdEOF	1	**EOF**: Moving past the end of an **rdoResultset** triggers the **RemoteData** control's Validation event on the last row, followed by a Reposition event on the invalid (**EOF**) row. At this point, the Move Next button on the **RemoteData** control is disabled.
rdAddNew	2	**AddNew**: Moving past the last row triggers the **RemoteData** control's Validation event to occur on the current row, followed by an automatic **AddNew**, followed by a Reposition event on the new row.

Remarks

If you set the **EOFAction** property to **rdAddNew**, once the user moves the current row pointer to **EOF** using the **RemoteData** control, the current row is positioned to a new row in the copy buffer. At this point you can edit the newly added row. If you make changes to the new row and the user subsequently moves the current row pointer using the **RemoteData** control, the row is automatically appended to the **rdoResultset**. If you don't make changes to this new row, and reposition the current row to another row, the new row is discarded. If you use the **RemoteData** control to position to another row while it is positioned over this new row, another new row is created.

When you use code to manipulate **rdoResultset** objects created with the **RemoteData** control, the **EOFAction** property has no effect it only takes effect when manipulating the **RemoteData** control with the mouse.

In situations where the **RemoteData** control **rdoResultset** is returned with no rows, or after the last row has been deleted, using the **rdAddNew** option for the **EOFAction** property greatly simplifies your code because a new row can always be edited as the current row.

Visual Basic Reference

Visual Studio 6.0

Bold Property

See Also Example Applies To

Returns or sets the font style of the **Font** object to either bold or nonbold.

Syntax

object.**Bold** [= *boolean*]

The **Bold** 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 the font style, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	Turns on bold formatting.
False	(Default) Turns off bold formatting.

Remarks

The **Font** object isn't directly available at design time. Instead you set the **Bold** property by selecting a control's **Font** property in the Properties window and clicking the Properties button. In the Font Style box of the Font dialog box, select either Bold or Bold Italic. At run time, however, you set **Bold** directly by specifying its setting for the **Font** object.

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 Basic: DataGrid Control

Visual Studio 6.0

Bookmark Property (DataGrid)

See Also Example Applies To

Returns or sets a bookmark for the specified row within a **RowBuffer** object in an unbound **DataGrid** control.

Syntax

object.**Bookmark** (*row*) [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
row	An integer specifying the row where the bookmark is placed. The range of this value can be from 0 to **RowCount** 1.
value	A variant representing the bookmark for the specified *row*.

Remarks

Use the value returned by the **Bookmark** property to save a reference to the current row that remains valid even after another row becomes current.

When you set the **Bookmark** property to a valid value in code, the row associated with that value becomes the current row, and the grid adjusts its display to bring the new current row into view if necessary.

The **Bookmark** property is defined as a Variant to accommodate user-defined bookmarks in unbound mode.

In the UnboundReadData event there may be multiple rows, so you must provide a bookmark for each row.

The UnboundWriteData event passes a bookmark to you to identify the row of data to be updated.

The UnboundAddData event passes a bookmark to you to identify the row of data to be added.

Note In unbound mode, setting the **Bookmark** property to itself will force the current row to be updated via the UnboundWriteData event.

Visual Basic: DataGrid Control

Bookmark Property (DataGrid) Example

In this example, when the user deletes a row in the unbound **DataGrid** control, the UnboundDeleteRow event is triggered, allowing you to manually delete the row from your data set, in this case, a simple array. The following code fragment shows how a bookmark is passed as an argument in the UnboundDeleteRow event to identify the row to be deleted.

```
Private Sub DataGrid1_UnboundDeleteRow(Bookmark As Variant)
  For i% = Bookmark + 1 To RowCount - 1
    For j% = 0 to MAXCOLS - 1
    UserData(j%, I% - 1) = UserData(j%, I%)
    Next j%
  Next I%
End Sub
```

Refer to the UnboundReadData event example for an example of assigning data to the **Bookmark** property.

Visual Basic: RDO Data Control

Visual Studio 6.0

Bookmark Property (Remote Data)

See Also Example Applies To

Returns or sets a bookmark that uniquely identifies the current row in an **rdoResultset** object. If you have a valid bookmark, you can use it to reposition the current row in an **rdoResultset**.

Syntax

object.**Bookmark** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Variant(string) expression that evaluates to a valid bookmark.

Remarks

When a non-forward-only-type **rdoResultset** object is created or opened, each of its rows already has a unique bookmark. You can save the bookmark for the current row by assigning the value of the **Bookmark** property to a variable declared as **Variant**. To quickly return to that row at any time after moving to a different row, set the **rdoResultset** object's **Bookmark** property to the value of that variable.

There is no limit to the number of bookmarks you can establish. To create a bookmark for a row other than the current row, move to the desired row and assign the value of the **Bookmark** property to a **Variant** variable that identifies the row.

To make sure the **rdoResultset** supports bookmarks, inspect the value of its **Bookmarkable** property before you use the **Bookmark** property. If **Bookmarkable** is **False**, the **rdoResultset** doesn't support bookmarks, and using the **Bookmark** property results in a trappable error. While a bookmark value might be returned when using a dynamic cursor, this value cannot always be trusted.

The value of the **Bookmark** property isn't guaranteed to be the same as a row number.

Note The **Bookmark** property doesn't apply to forward-only type **rdoResultset** objects.

Visual Basic: RDO Data Control

Visual Studio 6.0

Bookmarkable Property (Remote Data)

See Also Example Applies To

Returns a value that indicates whether an **rdoResultset** object supports bookmarks, which you can set using the **Bookmark** property.

Syntax

object.**Bookmarkable**

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

Return Values

The **Bookmarkable** property return values are:

Value	Description
True	The **rdoResultset** supports bookmarks.
False	The **rdoResultset** doesn't support bookmarks.

Remarks

To make sure an **rdoResultset** supports bookmarks, check the **Bookmarkable** property setting before you attempt to set or check the **Bookmark** property.

Visual Basic Reference

Visual Studio 6.0

BorderColor Property

See Also Example Applies To

Returns or sets the color of an object's border.

Syntax

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

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

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

Settings

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

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

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.

Visual Basic Reference

Visual Studio 6.0

BorderStyle Property

See Also Example Applies To

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

Syntax

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

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

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

Settings

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

Constant	Setting	Description
vbBSNone	0	None (no border or border-related elements).
vbFixedSingle	1	Fixed Single. Can include Control-menu box, title bar, Maximize button, and Minimize button. Resizable only using Maximize and Minimize buttons.
vbSizable	2	(Default) Sizable. Resizable using any of the optional border elements listed for setting 1.
vbFixedDouble	3	Fixed Dialog. Can include Control-menu box and title bar; can't include Maximize or Minimize buttons. Not resizable.
vbFixedToolWindow	4	Fixed ToolWindow. Displays a non-sizable window with a Close button and title bar text in a reduced font size. The form does not appear in the Windows taskbar.
vbSizableToolWindow	5	Sizable ToolWindow. Displays a sizable window with a Close button and title bar text in a reduced font size. The form does not appear in the Windows taskbar.

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

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

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

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

Remarks

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

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

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

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

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

Visual Basic Reference

Visual Studio 6.0

BorderStyle Property (ActiveX Controls)

See Also Example Applies To

Returns or sets the border style for an object.

Syntax

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

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

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

Settings

The **BorderStyle** property settings are:

Constant	Setting	Description
*Xx***None**	0	None (no border or border-related elements).
*Xx***FixedSingle**	1	Fixed Single.

Note Constants for ActiveX controls are prefaced by two letters which are specific to the control. For example, the Windows Common Controls use **ccNone**. In some cases, constants are totally changed. For example the MSChart control uses **vtBorderStyleNone**. However, the description remains the same unless indicated.

Visual Basic Reference

Visual Studio 6.0

BorderWidth Property

See Also Example Applies To

Returns or sets the width of a control's border.

Syntax

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

The **BorderWidth** 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 from 1 to 8192, inclusive.

Remarks

Use the **BorderWidth** and **BorderStyle** properties to specify the kind of border you want for a **Line** or **Shape** control. The following table shows the effect of **BorderStyle** settings on the **BorderWidth** property:

BorderStyle	Effect on BorderWidth
0	**BorderWidth** setting is ignored.
15	The border width expands from the center of the border; the height and width of the control are measured from the center of the border.
6	The border width expands inward on the control from the outside of the border; the height and width of the control are measured from the outside of the border.

If the **BorderWidth** property setting is greater than 1, the only effective settings of **BorderStyle** are 1 (Solid) and 6 (Inside Solid).

Visual Basic Reference

BorderWidth Property Example

This example uses two **ComboBox** controls to select different widths and styles for the borders of a **Shape** control. To try this example, paste the code into the Declarations section of a form that contains a **Shape** control and one **ComboBox** control. For the **ComboBox**, set **Style** = 2 and **Index** = 0 (to create a control array), and then press F5 and click the form.

```
Private Sub Form_Load ()
   Combo1(0).Width = 1440 * 1.5
   Load Combo1(1)
   Combo1(1).Top = Combo1(0).Top + Combo1(0).Height * 1.5
   Combo1(1).Visible = True
   For I = 0 To 6
      Combo1(0).AddItem "BorderStyle = " & I
   Next I
   For I = 1 To 10
      Combo1(1).AddItem "BorderWidth = " & I
   Next I
   Combo1(0).ListIndex = 1
   Combo1(1).ListIndex = 0
End Sub

Private Sub Combo1_Click (Index As Integer)
   If Index = 0 Then
      Shape1.BorderStyle = Combo1(0).ListIndex
   Else
      Shape1.BorderWidth = Combo1(1).ListIndex + 1
   End If
End Sub
```

Visual Basic Reference

Visual Studio 6.0

BottomMargin, TopMargin Properties

See Also Example Applies To

Returns or sets, in twips, the height of the bottom and top margins.

Syntax

object.**BottomMargin** [=*number*]

object.**TopMargin** [=*number*]

The **BottomMargin** and **TopMargin** properties syntax have these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
number	Optional. A numeric expression that specifies the height of the bottom margin or top margin.

Visual Basic: DataCombo/DataList Controls

Visual Studio 6.0

BoundColumn Property

See Also Example Applies To

Returns or sets the name of the source field in a **Recordset** object that is used to supply a data value to another **Recordset**.

Syntax

object.**BoundColumn** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string expression that specifies the name of a field in the **Recordset** created by the **Data** control specified by the **RowSource** property.

Remarks

Generally, when working with the **DataList** and **DataCombo** controls, you use two **Data** controls; one to fill the list as designated by the **Listfield** and **RowSource** properties and one to update a field in a database specified by the **DataSource** and **DataField** properties.

The **ListField** property designates the field used to fill the list. The second **Data** control, as designated by the **DataSource** property, manages a **Recordset** containing a field to be updated. Once the user chooses one of the items in the list, the field specified by the **BoundColumn** property is passed to the field in the second **Data** control, as designated by the **DataSource** and **DataField** properties. This way you can designate one field to fill the list, and another field (from the same **Recordset**) to pass as data to the **Recordset** designated by the **DataSource** and **DataField** properties when an item is selected.

If the field specified by the **BoundColumn** property can't be found in the **Recordset**, a trappable error occurs.

Data Type

String

Visual Basic: DataCombo/DataList Controls

Visual Studio 6.0

BoundText Property

See Also Example Applies To

Returns or sets the value of the field specified by the **BoundColumn** property.

Syntax

object.**BoundText** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string expression that specifies a data value.

Remarks

After a user makes a selection with a **DataList** or **DataCombo** control, the **BoundText** property contains the field value of the **BoundColumn** property. When an item in the list is selected, the item becomes available to the **Data** control specified by the **DataSource** property. The selection also appears in the text box portion of the **DataCombo** control where it can be edited. If the user enters a value in the text box portion, the list portion attempts to position to a matching item. If a match is found, the **BoundText** property value is set, based on the field value of the **BoundColumn** property. If no match is found, the **BoundText** property is set to Null.

You can use the **BoundText** property value to create a query that can then be used to find specific records:

```
Dim strQ As String
StrQ = "Select * From Products Where SupplierID = " & DataList1.BoundText
' Use the new query with an ADO Data Control to return a new recordset.
Adodc1.RecordSource = strQ
```

Positioning the **Data** control, specified by the **DataSource** property, to a new record sets the **BoundText** property to the value specified by **DataField**. The **DataList** or **DataCombo** control then searches the records in the list to see if the **BoundText** value matches the value of the field in the **BoundColumn** property. If a match is found, the record is highlighted in the list or placed in the text box portion of the **DataCombo** control.

Data Type

String

Visual Basic: MSComm Control

Visual Studio 6.0

Break Property

See Also Example Applies To

Sets or clears the break signal state. This property is not available at design time.

Syntax

object.**Break** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression specifying whether the break signal state is set, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	Sets the break signal state.
False	Clears the break signal state.

Remarks

When set to **True**, the **Break** property sends a break signal. The break signal suspends character transmission and places the transmission line in a break state until you set the **Break** property to **False**.

Typically, you set the break state for a short interval of time, and *only* if the device with which you are communicating requires that a break signal be set.

Data Type

Boolean

Visual Basic: MSComm Control

Break Property Example

The following example shows how to send a break signal for a tenth of a second:

```
' Set the Break condition.
MSComm1.Break = True
' Set duration to 1/10 second.
Duration! = Timer + .1
' Wait for the duration to pass.
Do Until Timer > Duration!
   Dummy = DoEvents()
Loop
' Clear the Break condition.
MSComm1.Break = False
```

Visual Basic Reference

Visual Studio 6.0

Browsable Property

See Also Example Applies To

Returns or sets the Browsable attribute associated with a **Member** object.

Syntax

object.**Browsable**

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

Visual Basic Reference

Visual Studio 6.0

BrowserType Property

See Also Example Applies To

Returns the Active Server Pages **BrowserType** object.

Syntax

object.**BrowserType**

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

Remarks

A WebClass uses the **BrowserType** object to determine the attributes of the users browser and make processing decisions based on those attributes.

See the Active Server Pages documentation for details of the properties, methods, and events for the **BrowserType** object.

Visual Studio 6.0

Visual Basic: MSChart Control

Brush Property

See Also Example Applies To

Returns a reference to a **Brush** object that describes the fill type used to display a chart element.

Syntax

object.**Brush**

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

Visual Basic: MSChart Control

Brush Object Example

The following example sets a bold vertical line pattern for the chart backdrop using the **Brush** object.

```
Private Sub Command1_Click()
   ' Sets Backdrop to Fill - Brush Style.
   MSChart1.Backdrop.Fill.Style = VtFillStyleBrush
   ' Sets a pattern for the chart backdrop using the
   ' Brush object.
   With MSChart1.Backdrop.Fill.Brush
      .Style = VtBrushStylePattern
      .Index = VtBrushPatternBoldVertical
   ' Sets Pattern to Bold Vertical lines.
      .FillColor.Set 255, 0, 0   ' Fill Color = Red.
      .PatternColor.Set 0, 0, 255   ' Pattern Color =
                                ' Blue.
   End With
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

BuddyProperty Property

See Also Example Applies To

Sets or returns the property used to synchronize the **UpDown** control with its buddy control.

Syntax

object.**BuddyProperty** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A variant value that specifies a property of the control specified by the **BuddyControl** property. If no value is specified, the default property for the control is used.

Remarks

If the **SyncBuddy** property is **True**, the control will synchronize its **Value** property with the property specified by the **BuddyProperty** property.

The **BuddyProperty** can be set at design time by selecting the property out of a drop down list in the Properties window.

The **BuddyControl** property must be set before setting the **BuddyProperty** property, or an error results.

Visual Basic: Page Designer

Visual Studio 6.0

BuildFile Property

See Also Example Applies To

Sets the path and file name to which the designer should save the compiled HTML for this designer. Not available at run time.

Remarks

This property is initially set to the same path as the **SourceFile** property. You can change this path to any location to which you want to save the compiled HTML file.

You must enter an absolute path to the build location, indicating the full drive letter and directory structure in which the compiled file will be placed. If you later share your project with another developer who has a different directory structure or drive name than you originally used, the second developer will need to modify the value of the **BuildFile** property to reflect the location of the compiled HTML files on his machine. Run-time users are not affected by this restriction.

Visual Basic Extensibility Reference

Visual Studio 6.0

BuildFileName Property

See Also Example Applies To

Sets or returns the executable or DLL name that will be used when the project is built.

Syntax

object.**BuildFileName**

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

Visual Basic Extensibility Reference

Visual Studio 6.0

BuiltIn Property

See Also Example Applies To Specifics

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

Return Values

The **BuiltIn** property returns these values:

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

Visual Basic Extensibility Reference

BuiltIn Property Example

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

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

Visual Basic: RichTextBox Control

Visual Studio 6.0

BulletIndent Property

See Also Example Applies To

Returns or sets the amount of indent used in a **RichTextBox** control when **SelBullet** is set to **True**.

Syntax

object.**BulletIndent** [= *integer*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
integer	An integer that determines the amount of indent. These properties use the scale mode units of the **Form** object containing the **RichTextBox** control.

Remarks

The **BulletIndent** property returns **Null** if the selection spans multiple paragraphs with different margin settings.

Visual Basic: DataGrid Control

Visual Studio 6.0

Button Property (Column Object)

See Also Example Applies To

Sets or returns a value that determines whether a button is displayed within the current cell.

Syntax

object.**Button** [= *value*]

The **Button** 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 if a button is displayed within the current cell, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	A button will be displayed in the upper right corner of the current cell at run time.
False	(Default) No button will be displayed.

Remarks

Typically, you enable the column button when you want to drop down a control (such as the built-in combo box, a bound list box, or even another **DataGrid** control) for editing or data entry. When the button in the current cell is clicked, the ButtonClick event will be fired. You can then write code to drop down the desired control from the cell.

Visual Basic: Multimedia MCI Control

Visual Studio 6.0

ButtonEnabled Property (Multimedia MCI Control)

See Also Example Applies To

Determines if a button in the control is enabled or disabled (a disabled button appears dimmed).

Syntax

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

Remarks

The effects of the *Button***Enabled** properties are superseded by the **Enabled** and **AutoEnable** properties. Individual *Button***Enabled** properties enable or disable the associated buttons in the **Multimedia MCI** control when the **Multimedia MCI** control is enabled (**Enabled** property set to **True**) and the **AutoEnable** property is turned off (set to **False**).

For this property, *Button* may be any of the following: Back, Eject, Next, Pause, Play, Prev, Record, Step, or Stop. For example, to disable the Play button, use:

```
[form.]MMControl.PlayEnabled = False
```

To check whether the Record button is enabled, use:

```
If [form.]MMControl.RecordEnabled Then ..
```

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

Setting	Description
False	(Default) Disables (dims) the button specified by *Button*. This button's function is not available in the control.
True	Enables the button specified by *Button*. This button's function is available in the control.

Data Type

Integer (Boolean)

Visual Basic: Windows Controls

Visual Studio 6.0

ButtonHeight, ButtonWidth Properties

See Also Example Applies To

Return or set the height and width of a **Toolbar** control's buttons.

Syntax

object.**ButtonHeight** [= *number*]

object.**ButtonWidth** [= *number*]

The **ButtonHeight, ButtonWidth** properties syntax has these parts:

Part	Description
object	An object expression that evaluates to a **Toolbar** control.
number	A numeric expression specifying the dimensions of all buttons on the control that have the Button, Check, or ButtonGroup style.

Remarks

ButtonHeight and **ButtonWidth** use the scale unit of the Toolbar control's container. The scale unit is determined by the **ScaleMode** property of the container.

By default, the **ButtonWidth** and **ButtonHeight** properties are automatically updated to accommodate the string in the **Caption** property or image in the **Image** property of the **Button** object.

Visual Basic: Windows Controls

Visual Studio 6.0

ButtonMenus Property

See Also Example Applies To

Returns a reference to the **ButtonMenus** collection. Available at run time only.

Syntax

object.**ButtonMenus**

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

Buttons Property

See Also Example Applies To

Returns a reference to a **Toolbar** control's collection of **Button** objects.

Syntax

object.**Buttons**

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

Remarks

You can manipulate **Button** objects using standard collection methods (for example, the **Add** and **Remove** methods). Each element in the collection can be accessed by its index, the value of the **Index** property, or by a unique key, the value of the **Key** property.

Visual Basic: Multimedia MCI Control

Visual Studio 6.0

ButtonVisible Property (Multimedia MCI Control)

See Also Example Applies To

Determines if the specified button is displayed in the control.

Syntax

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

Remarks

The effects of the *Button***Visible** properties are superseded by the **Visible** property. Individual *Button***Visible** properties display and hide the associated buttons in the **Multimedia MCI** control when the **Multimedia MCI** control is visible (**Visible** property set to **True**). If the **Multimedia MCI** control is invisible, these properties are not used.

For this property, *Button* may be any of the following: Back, Eject, Next, Pause, Play, Prev, Record, Step, or Stop.

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

Setting	Description
False	Does not display the button specified by *Button*. This button's function is not available in the control.
True	(Default) Displays the button specified by *Button*.

Data Type

Integer (Boolean)

Visual Basic Reference

Visual Studio 6.0

BytesMax Property

See Also Example Applies To

Returns the estimated maximum number of bytes to be read.

Syntax

object.**BytesMax**

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

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

Remarks

The **BytesMax** property returns a long integer. The value of **BytesMax** can change during the download if a more accurate estimate can be made. The value can be 0 for some downloads if the server can't determine the download size. For example, .htm pages don't always provide this information.

Visual Basic Reference

Visual Studio 6.0

BytesRead Property

See Also Example Applies To

Returns the total number of bytes that have been currently read or downloaded.

Syntax

object.**BytesRead**

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

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

Remarks

The **BytesRead** property returns a long integer.

Visual Basic: Winsock Control

Visual Studio 6.0

BytesReceived Property

See Also Example Applies To

Returns the amount of data received (currently in the receive buffer). Use the **GetData** method to retrieve data.

Read-only and unavailable at design time.

Syntax

object.**BytesReceived**

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

Return Value

Long