This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

MainWindow Property

See Also Example Applies To Specifics

Returns a **Window** object representing the main window of the Visual Basic development environment. Read-only.

Remarks

You can use the **Window** object returned by the **MainWindow** property to add or remove docked windows. You can also use the **Window** object returned by the **MainWindow** property to maximize, minimize, hide, or restore the main window of the Visual Basic development environment.

© 2017 Microsoft

Visual Basic Extensibility Reference

MainWindow Property Example

The following example uses the **MainWindow** property to return the **Window** object representing the main window, and then prints the caption of the main window.

```
Debug.Print Application.VBE.MainWindow.Caption
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Major Property

See Also Example Applies To

Returns or sets the major release number of the project. Read only at run time.

Syntax

object.**Major**

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

Remarks

The value of the **Major** property is in the range from 0 to 9999.

This property provides version information about the running application.

You can set this property at design time in the Major box in the Make tab of the Project Properties dialog box.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

Major Property

See Also Example Applies To Specifics

Returns a Long containing the major version number of the referenced type library. Read-only.

Remarks

The number returned by the **Major** property corresponds to the major version number stored in the type library to which you have set the reference.

© 2017 Microsoft

Visual Basic Extensibility Reference

Major Property Example

The following example uses the **Major** property to return the major version number of the specified **Reference** object in a particular project.

```
Debug.Print Application.VBE.VBProjects(1).References(1).Major
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

MajorDivision Property

See Also Example Applies To

Returns or sets the number of major divisions displayed on the axis.

Syntax

object.**MajorDivision** [= *num*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
num	Integer. Number of divisions.

Remarks

If this property is set, then the **ValueScale** object's **Auto** property is automatically set to **False**.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

MajorPen Property

See Also Example Applies To

Returns a reference to a **Pen** object that describes the appearance of the major axis grid lines.

Syntax

object.**MajorPen**

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

Marker Property

See Also Example Applies To

Returns a reference to a **Marker** object that describes the icon used to identify a data point on a chart.

Syntax

object.**Marker**

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

MarqueeStyle Property

See Also Example Applies To

Sets or returns the Marquee style for the **DataGrid** control or **Split** object.

Syntax

object.**MarqueeStyle** [= *value*]

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

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

Settings

The settings for *value* are:

Constant	Value	Description
dbgDottedCellBorder	0	The current cell within the current row will be highlighted by drawing a dotted border around the cell. In Microsoft Windows terminology, this is usually called a focus rectangle.
dbgSolidCellBorder	1	The current cell within the current row will be highlighted by drawing a solid box around the current cell. This is more visible than the dotted cell border, especially when 3-D divider properties are used for the grid.
dbgHighlightCell	2	The entire current cell will be highlighted by inverting the colors within the cell. This provides a very distinctive block-style highlight for the current cell.
dbgHighlightRow	3	The entire row containing the current cell will be highlighted by inverting the colors within the row. In this mode, it is not possible to visually determine which cell is the current cell, only the current row. When the grid or split is not editable, this setting is often preferred, since cell position is then irrelevant.
dbgHighlightRowRaiseCell	4	The entire row will be highlighted. The current cell within the row will be "raised" so that it appears distinctive. This setting doesn't appear clearly with all

		background color and divider settings. The best effect is achieved by using 3-D dividers and a light gray background.
dbgNoMarquee	5	The marquee will not be shown. This setting is useful for cases where the current row is irrelevant, or where you don't want to draw the user's attention to the grid until necessary.
dbgFloatingEditor	6	The current cell will be highlighted by a floating text editor window with a blinking caret (as in Microsoft Access). This is the default setting.

Remarks

If a grid contains multiple splits, then setting its **MarqueeStyle** property has the same effect as setting the **MarqueeStyle** property of each split individually.

Note If the floating editor marquee setting is in effect and the current cell contains radio buttons or graphics, then a dotted focus rectangle will be displayed.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MaskedEdit Control

Visual Studio 6.0

Mask Property

See Also Example Applies To

Determines the input mask for the control.

Syntax

[*form.*]**MaskedEdit.Mask** [= *string$*]

Remarks

You can define input masks at both design time and run time. However, the following are examples of standard input masks that you may want to use at design time. The control can distinguish between numeric and alphabetic characters for validation, but cannot check for valid content, such as the correct month or time of day.

Mask	Description
Null String	(Default) No mask. Acts like a standard text box.
##-???-##	Medium date (US). Example: 20-May-92
##-##-##	Short date (US). Example: 05-20-92
##:## ??	Medium time. Example: 05:36 AM
##:##	Short time. Example: 17:23

The input mask can consist of the following characters.

Mask character	Description
#	Digit placeholder.
.	Decimal placeholder. The actual character used is the one specified as the decimal placeholder in your international settings. This character is treated as a literal for masking purposes.
,	Thousands separator. The actual character used is the one specified as the thousands separator in your international settings. This character is treated as a literal for masking purposes.

:	Time separator. The actual character used is the one specified as the time separator in your international settings. This character is treated as a literal for masking purposes.
/	Date separator. The actual character used is the one specified as the date separator in your international settings. This character is treated as a literal for masking purposes.
\	Treat the next character in the mask string as a literal. This allows you to include the '#', '&', 'A', and '?' characters in the mask. This character is treated as a literal for masking purposes.
&	Character placeholder. Valid values for this placeholder are ANSI characters in the following ranges: 32-126 and 128-255.
>	Convert all the characters that follow to uppercase.
<	Convert all the characters that follow to lowercase.
A	Alphanumeric character placeholder (entry required). For example: a z, A Z, or 0 9.
a	Alphanumeric character placeholder (entry optional).
9	Digit placeholder (entry optional). For example: 0 9.
C	Character or space placeholder (entry optional). This operates exactly like the & placeholder, and ensures compatibility with Microsoft Access.
?	Letter placeholder. For example: a z or A Z.
Literal	All other symbols are displayed as literals; that is, as themselves.

When the value of the **Mask** property is an empty string (""), the control behaves like a standard text box control. When an input mask is defined, underscores appear beneath every placeholder in the mask. You can only replace a placeholder with a character that is of the same type as the one specified in the input mask. If you enter an invalid character, the masked edit control rejects the character and generates a ValidationError event.

Note When you define an input mask for the **Masked Edit** control and you tab to another control, the ValidationError event is generated if there are any invalid characters in the **Masked Edit** control.

Data Type

String

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

MaskColor Property

See Also Example Applies To

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

Syntax

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

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

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

Remarks

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

If the system colors change, then the color which is transparent will change, making the look of your picture unpredictable. It is good programming practice to use non-system colors.

© 2017 Microsoft

Visual Basic: Windows Controls

MaskColor Property Example

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

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

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

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

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

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

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

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

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MaskColor Property (UserControl Object)

See Also Example Applies To

Returns or sets the color that determines the transparent region of the bitmap assigned to the **MaskPicture** property of a **UserControl** object whose **BackStyle** property is set to 0 (Transparent).

Syntax

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

The **MaskColor** 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 color to be used as a mask, 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
RGB colors	Colors specified by using the Color palette or by using the **RGB** or **QBColor** functions in code.

Remarks

When a bitmap is assigned to the **MaskPicture** property of a **UserControl** whose **BackStyle** property is set to 0 (Transparent), the control becomes transparent wherever it is covered by areas of the bitmap that match the **MaskColor** property.

Mouse events that occur over the transparent areas are received by the container or by controls that would otherwise be covered by the **UserControl**.

If there is no bitmap assigned to the **MaskPicture** property, or if the **BackStyle** property of the **UserControl** is not set to 0 (Transparent), then setting the **MaskColor** property has no effect.

For further details, see the **MaskPicture** property of the **UserControl** object.

Note Although **MaskColor** accepts the system color constants listed in the Visual Basic (VB) object library in the Object Browser, as described in Help for the **BackColor** and **ForeColor** properties, this is only useful if the **MaskPicture** bitmap contains a system color.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MaskPicture Property (UserControl Object)

See Also Example Applies To

Returns or sets the bitmap that, combined with the **MaskColor** property, determines the transparent and visible regions of a **UserControl** object whose **BackStyle** property is set to 0 (Transparent).

Syntax

object.**MaskPicture** [= *picture*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
picture	A graphic image of one of the types described in Settings.

Settings

The settings for *picture* are:

Setting	Description
Nothing	(Default) No picture. At design time, highlight the property value and press the Delete key; "(None)" will appear in the Properties window. At run time, assign the return value of the **LoadPicture** function with no filename, or use the **Set** statement to assign the value **Nothing** to the property.
(Bitmap, DIB, GIF, or JPEG)	Specifies an image-type bitmap. At design time, you can use the Properties window to enter a string expression specifying a file containing a graphic. At run time, you can set this property using a **Picture** object, using another object's **Picture** property, or using the **LoadPicture** function on an image-type bitmap file.

Important This feature is supported only for image-type bitmaps, such as GIF, JPEG, and DIB. It is not supported for Windows metafiles, icons, or cursors.

Remarks

When a bitmap is assigned to the **MaskPicture** property of a **UserControl** whose **BackStyle** property is set to 0 (Transparent), the control becomes transparent wherever it is covered by areas of the bitmap that match the **MaskColor**

property.

Mouse events that occur over the transparent areas are received by the container or by controls that would otherwise be covered by the **UserControl**.

The non-transparent parts of the **MaskPicture** bitmap are painted with the color specified by the **UserControl**'s **BackColor** property. These areas, which need not be contiguous, define the clipping region for drawing on the **UserControl**. That is, any drawing that is done on the surface of the **UserControl** is clipped to the non-transparent parts of the **MaskPicture** bitmap.

Important The bitmap assigned to the **MaskPicture** property is only used to define the transparent region and clipping region for the **UserControl**. The bitmap is never displayed. To display a bitmap on top of the clipping region defined by **MaskPicture** and **MaskColor**, you can assign the bitmap to the **Picture** property of the **UserControl**, or draw it on the **UserControl** using **PaintPicture** in the **UserControl**'s Paint event.

You can create an animated clipping region for your control by drawing on a hidden **PictureBox**, and transferring the resulting image to the **MaskPicture** property of the **UserControl**.

Note When you set the **MaskPicture** property at design time, the graphic is saved and loaded with the .ctl and .ctx files that define the **UserControl**. If you make the project into a control component (.ocx file), the file contains the image. When you load a graphic at run time, by contrast, the graphic isn't saved with the component.

For more information, see "Giving Your Control a Transparent Background," in "Building ActiveX Controls" in "Creating ActiveX Components" in the *Component Tools Guide*.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataCombo/DataList Controls

Visual Studio 6.0

MatchedWithList Property

See Also Example Applies To

Returns **True** if the current content of the **BoundText** property matches one of the records in the list portion of the control.

Syntax

object.**MatchedWithList**

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

Return Values

The **MatchedWithList** property return values are:

Value	Description
True	The content of the **BoundText** property matches one of the records in the list.
False	The contents of the **BoundText** property does not match any of the records in the list.

Remarks

When you enter a value in the text portion of the **DataCombo** control, the **MatchedWithList** property is set to **True** if the value entered is one of the items shown in the list. Moving the **Data** control specified by the **DataSource** property of the **DataCombo** or **DataList** control also sets the **MatchedWithList** property to **True** if the **BoundText** value matches one of the records in the list. In this case, the record is highlighted.

Data Type

Boolean

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataCombo/DataList Controls

Visual Studio 6.0

MatchEntry Property

See Also Example Applies To

Returns or sets a value indicating how the **DataCombo** or **DataList** control performs searches based on user input.

Syntax

object.**MatchEntry** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A constant or value that defines the behavior of a control when it has focus and the user enters one or more characters, as described in Settings.

Settings

The settings for *value* are:

Setting	Value	Description
dblBasicMatching	0	Basic Matching: (Default) The control searches for the next match for the character entered using the first letter of entries in the list. Repeatedly typing the same letter cycles through all of the entries in the list beginning with that letter.
dblExtendedMatching	1	Extended Matching: The control searches for an entry matching all characters entered. The search is done as characters are being typed, further refining the search.

Remarks

When the **MatchEntry** property is set to **dblExtendedMatching** and the user enters a backspace or waits more than a few seconds, the matching string is reset.

Data Type

Integer

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

Max Property (MSChart)

See Also Example Applies To

Returns a reference to a **Coor** object that specifies the ending corner of a rectangle.

Syntax

object.**Max**

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

Max Property

See Also Example Applies To

Sets or returns the maximum value of the scroll range for the **UpDown** control.

Syntax

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

The **Max** 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 value that specifies the maximum value, as described below. The setting for *value* can be a negative number.

Remarks

Pressing the up or right arrow normally causes the **UpDown** control to increase the **Value** property, however, if the **Max** property is less than the **Min** property, the **UpDown** control operates in the reverse direction.

Pressing the up or right arrow always causes the **Value** property to approach the **Max** value. Pressing the down or left arrow always causes the **Value** property to approach the **Min** value.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Max, Min Properties (ActiveX Controls)

See Also Example Applies To

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

Syntax

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

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

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

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

Settings

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

- **Max** 32,767.
- **Min** 0.

Remarks

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

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

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

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: CommonDialog Control

Visual Studio 6.0

Max, Min Properties (CommonDialog)

See Also Example Applies To

You can set the **Max** and **Min** properties for the following:

- Font dialog box return or set the smallest and largest font sizes displayed in the Size list box.
- Print dialog box return or set the minimum and maximum allowed values for the print range.

Syntax

object.**Min** [= *points*]

object.**Max** [= *points*]

object.**Min** [= *number*]

object.**Max** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
points	A numeric expression specifying the smallest and largest font sizes.
number	A numeric expression specifying the minimum and maximum page numbers.

Remarks

- With the Font dialog box, the **cdlCFLimitSize** flag must be set before using these properties.
- With the Print dialog box, the **Min** property determines the smallest number the user can specify in the From text box. The **Max** property determines the largest number the user can specify in the To text box.

Data Type

Integer

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Max, Min Properties (Scroll Bar)

See Also Example Applies To

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

Syntax

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

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

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

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

Settings

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

- **Max** 32,767.
- **Min** 0.

Remarks

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

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

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

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MaxButton Property

See Also Example Applies To

Returns a value indicating whether a form has a Maximize button.

Syntax

object.**MaxButton**

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

Settings

The **MaxButton** property settings are:

Setting	Description
True	(Default) The form has a Maximize button.
False	The form doesn't have a Maximize button.

Remarks

A Maximize button enables users to enlarge a form window to full-screen size. To display a Maximize button, you must also set the form's **BorderStyle** property to either 1 (Fixed Single), 2 (Sizable), or 3 (Fixed Double).

A Maximize button automatically becomes a Restore button when a window is maximized. Minimizing or restoring a window automatically changes the Restore button back to a Maximize button.

The settings you specify for the **MaxButton**, **MinButton**, **BorderStyle**, and **ControlBox** properties aren't reflected in the form's appearance until run time.

Note Maximizing a form at run time generates a Resize event. The **WindowState** property reflects the current state of the window. If you set the **WindowState** property to 2 (Maximized), the form is maximized independently of whatever settings are in effect for the **MaxButton** and **BorderStyle** properties.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

MaxDate, MinDate Properties

See Also Example Applies To

Returns or sets the first and last date allowed by the calendar.

Syntax

object.**MaxDate** [= *date*]

object.**MinDate** [= *date*]

The **MaxDate** and **MinDate** property syntaxs have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
date	A date expression that evaluates to a valid date.

Remarks

The **MaxDate** property is used to set the upper boundary of the calendar.

The **MinDate** property is used to set the lower boundary of the calendar.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: CommonDialog Control

Visual Studio 6.0

MaxFileSize Property

See Also Example Applies To

Returns or sets the maximum size of the filename opened using the **CommonDialog** control.

Syntax

object.**MaxFileSize** [= *value*]

The **MaxFileSize** 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 maximum size of the filename in bytes. The range for this property is 1 32K. The default is 256.

Remarks

The **MaxFileSize** property allocates memory to store the actual names of the selected file or files. When using the **cdlOFNAllowMultiselect** flag, you may want to increase the size of the **MaxFileSize** property to allow enough memory for the selected file names.

Data Type

Integer

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

Maximum Property

See Also Example Applies To

Returns or sets the highest or ending value on the chart value axis.

Syntax

object.**Maximum** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	Double. The highest axis value.

Remarks

If this property is set, then the **ValueScale** object's **Auto** property is automatically set to **False**.

The **Maximum** property should be set before the **Minimum** property to avoid a chart display error.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MaxLength Property

See Also Example Applies To

Returns or sets a value indicating whether there is a maximum number of characters that can be entered in the **TextBox** control and, if so, specifies the maximum number of characters that can be entered.

Note In DBCS (double-byte character set) systems, each character can take up to two bytes instead of only one, which limits the number of characters you can enter.

Syntax

object.**MaxLength** [= *value*]

The **MaxLength** 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 maximum number of characters a user can enter in a **TextBox** control. The default for the **MaxLength** property is 0, indicating no maximum other than that created by memory constraints on the user's system for single-line **TextBox** controls and a maximum of approximately 32K for multiple-line **TextBox** controls. Any number greater than 0 indicates the maximum number of characters.

Remarks

Use the **MaxLength** property to limit the number of characters a user can enter in a **TextBox**.

If text that exceeds the **MaxLength** property setting is assigned to a **TextBox** from code, no error occurs; however, only the maximum number of characters is assigned to the **Text** property, and extra characters are truncated. Changing this property doesn't affect the current contents of a **TextBox** but will affect any subsequent changes to the contents.

© 2017 Microsoft

Visual Basic Reference

MaxLength Property Example

This example uses a numeric value in one TextBox control to limit the length of text in another TextBox control. To try this example, paste the code into the Declarations section of a form that contains two TextBox controls. Make Text1 fairly large, and then press F5. Enter a number into Text2 and text into Text1.

```
Private Sub Text1_Change ()
   Text1.MaxLength = Text2.Text
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: RichTextBox Control

Visual Studio 6.0

MaxLength Property (RichTextBox Control)

See Also Example Applies To

Returns or sets a value indicating whether there is a maximum number of characters a **RichTextBox** control can hold and, if so, specifies the maximum number of characters.

Syntax

object.**MaxLength** [= *long*]

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

Part	Description
object	An object expression that evaluates to a **RichTextBox** control.
long	A long integer specifying the maximum number of characters a user can enter in the control. The default for the **MaxLength** property is 0, indicating no maximum other than that created by memory constraints on the user's system. Any number greater than 0 indicates the maximum number of characters.

Remarks

Use the **MaxLength** property to limit the number of characters a user can enter in a **RichTextBox**.

If text that exceeds the **MaxLength** property setting is assigned to a **RichTextBox** from code, no error occurs; however, only the maximum number of characters is assigned to the **Text** property, and extra characters are truncated. Changing this property doesn't affect the current contents of a **RichTextBox**, but will affect any subsequent changes to the contents.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MaxRecords Property

See Also Example Applies To

Returns or sets the maximum number of records that can be retrieved from the data source.

Syntax

object.**MaxRecords** [=*number*]

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

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
number	Integer. A numeric expression that specifies the number of records that can be retrieved from the data source.

Remarks

This property corresponds to the ADO Recordset MaxRecords property.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: RDO Data Control

Visual Studio 6.0

MaxRows Property (Remote Data)

See Also Example Applies To

Returns or sets a value indicating the maximum number of rows to be returned from a query or processed in an action query.

Syntax

object.**MaxRows** [= *value*]

The **MaxRows** 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 Settings.

Settings

The setting for *value* ranges from 0 to any number. If *value* is set to 0, no limit is placed on the number of rows returned (default). Setting *value* to a negative number is invalid and is automatically reset to 0.

Remarks

The **MaxRows** property limits the number of rows processed by the remote server. When **MaxRows** is set to a value greater than 0, only '*n*' rows are processed. When executing a query that returns rows, it means that only the first '*n*' rows are returned. When executing an action query, it means that only the first '*n*' rows are updated, inserted or deleted.

This property is useful in situations where limited resources prohibit management of large numbers of result set rows. By setting **MaxRows** to 1 on an action query, you can be assured that no more than one row will be affected by the operation.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

MaxSelCount Property

See Also Example Applies To

Returns or sets the maximum number of contiguous days that can be selected at once.

Syntax

object.**MaxSelCount** [= *number*]

The **MaxSelCount** 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 the number of days that can be selected at once.

Remarks

The **MaxSelCount** property is valid only when the **MultiSelect** property is to **True**. Additionally, the **MaxSelCount** property must be set to a value that is greater than the difference between the **SelStart** and **SelEnd** properties. For example, given a selection of 9/15 to 9/18, `MonthView.SelEnd - MonthView.SelStart` = 3. However, four days are actually selected; thus **MaxSelCount** must be set to 4.

The default of the property is one week (7 days).

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MDIChild Property

See Also Example Applies To

Returns or sets a value indicating whether a form is displayed as an MDI child form inside an MDI form. Read only at run time.

Syntax

object.**MDIChild**

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

Settings

The **MDIChild** property settings are:

Setting	Description
True	The form is an MDI child form and is displayed inside the parent MDI form.
False	(Default) The form isn't an MDI child form.

Remarks

Use this property when creating a multiple-document interface (MDI) application. At run time, forms with this property set to **True** are displayed inside an MDI form. An MDI child form can be maximized, minimized, and moved, all inside the parent MDI form.

When working with MDI child forms, keep the following in mind:

- At run time, when an MDI child form is maximized, its caption is combined with that of the parent MDI form.
- At design time, an MDI child form is displayed like any other form because the form is displayed inside the parent form only at run time. An MDI child form's icon in the Project window is different from icons for other kinds of forms.
- MDI child forms can't be modal.
- The initial size and placement of MDI child forms are controlled by the Microsoft Windows operating environment unless you specifically set them in the Load event procedure.
- If an MDI child form is referenced before the parent is loaded, the parent MDI form is automatically loaded. However, if the parent MDI form is referenced before loading an MDI child form, the child form isn't loaded.

Note All MDI child forms have sizable borders, a Control-menu box, and Minimize and Maximize buttons, regardless of the settings of the **BorderStyle**, **ControlBox**, **MinButton**, and **MaxButton** properties.

Any reference to an **MDIForm** object, including reading or setting properties, causes the form to load and become visible.

© 2017 Microsoft

Visual Basic Reference

MDIChild Property Example

This example creates a second instance of an MDI child form within an **MDIForm** object. To try this example, set the **MDIChild** property to **True** on Form1, and then create an **MDIForm** object with the Add MDI Form command on the Project menu. Paste the code into the Declarations section of the **MDIForm**, and then press F5 to run the program.

```
Private Sub MDIForm_Load ()
   Dim NewForm As New Form1   ' Declare new form.
   NewForm.Show   ' Show new form.
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Members Property

See Also Example Applies To

Contains identifiers that have module-level scope and can be considered properties, methods, or events of the specified **CodeModule** object.

Syntax

object.**Members**

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

MergeCells Property

See Also Example Applies To

Returns or sets a value that determines whether cells with the same contents should be grouped in a single cell spanning multiple rows or columns.

Syntax

object.**MergeCells** [=*value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer or constant that specifies the grouping (merging) of cells, as specified in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
flexMergeNever	0	Never. The cells containing identical content are not grouped. This is the default.
flexMergeFree	1	Free. Cells with identical content always merge.
flexMergeRestrictRows	2	Restrict Rows. Only adjacent cells (to the left of the current cell) within the row containing identical content merge.
flexMergeRestrictColumns	3	Restrict Columns. Only adjacent cells (to the top of the current cell) within the column containing identical content merge.
flexMergeRestrictBoth	4	Restrict Both. Only adjacent cells within the row (to the left) or column (to the top) containing identical content merge.

Remarks

The ability to merge cells enables you to present data in a clear, concise manner. You can use cell merging in conjunction with the sorting and column order functions of the **MSHFlexGrid**.

To use the cell merging capabilities of the **MSHFlexGrid**:

- Set **MergeCells** to a value other than zero. (The difference between the settings is explained in the example.)
- Set the **MergeRow** and **MergeCol** array properties to **True** for the rows and columns to be merged.

When using the cell merging capabilities, the **MSHFlexGrid** merges cells with identical content. The merging is automatically updated whenever the cell content changes.

When **MergeCells** is set to a value other than 0 (Never), selection highlighting is automatically turned off. This is done to speed up repainting, and because selection of ranges containing merged cells may lead to unexpected results.

© 2017 Microsoft

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

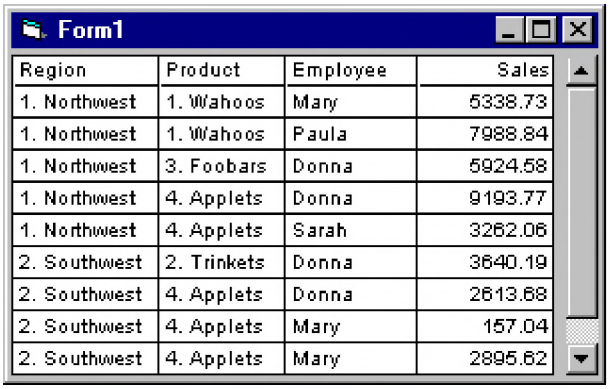
MergeCells Property Example

The following example shows the basic **MergeCells** property.

No Merging

```
MergeCells =0
MergeRow(0) =True
MergeRow(1) =True
MergeRow(2) =True
MergeRow(3) =False
```

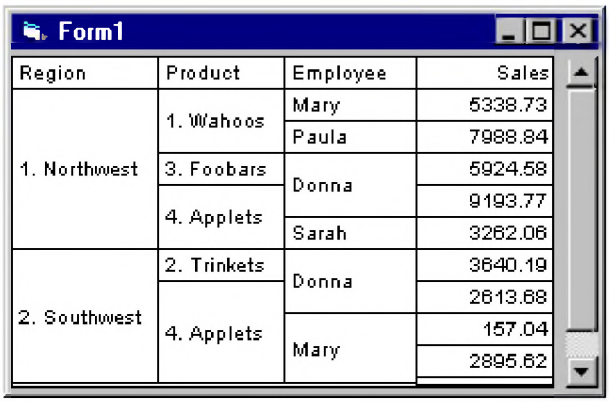
This is the regular view.



Free Merging

```
MergeCells =1
MergeRow(0) =True
MergeRow(1) =True
MergeRow(2) =True
MergeRow(3) =False
```

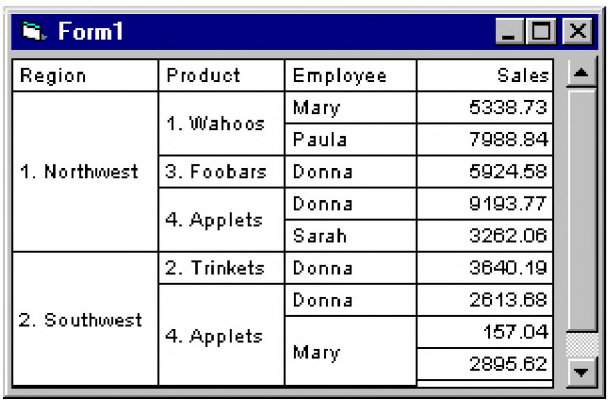
Notice how the third employee cell (Donna) merges across products to its left and across sales to its right.



Restricted Merging

```
MergeCells =2
MergeRow(0) =True
MergeRow(1) =True
MergeRow(2) =True
MergeRow(3) =False
```

Notice how the third employee cell (Donna) no longer merges across sales.



© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

MergeCol, MergeRow Properties

See Also Example Applies To

Returns or sets a value that determines which rows and columns can have their contents merged. These properties must be **True** to use the **MergeCells** property.

Syntax

object.**MergeCol(***number***)** [=*Boolean*]
object.**MergeRow(***number***)** [=*Boolean*]

Syntax for the **MergeCol** and **MergeRows** properties has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A Long value that specifies the column or row in the **MSHFlexGrid**.
Boolean	A Boolean that specifies whether merging occurs when adjacent cells display identical content.

Settings

The settings for *Boolean* are:

Part	Description
True	When adjacent cells display identical content, the rows merge to the left or the columns merge to the top.
False	When adjacent cells display identical content, no cells merge. This is the default for **MergeCol** and **MergeRow**.

Remarks

If the **MergeCells** property is set to a nonzero value, adjacent cells with identical values are merged only if they are in a row with the **MergeRow** property set to **True** or in a column with the **MergeCol** property set to **True**.

For details on the merging functionality of the **MSHFlexGrid**, see the **MergeCells** property.

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MessageReflect Property

See Also Example Applies To

Returns a boolean value stating whether the control container handles message reflection automatically.

Syntax

object.**MessageReflect**

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

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

Settings

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

Setting	Description
True	The container for the control will reflect messages.
False	The container for the control cannot reflect messages. If the container does not implement this ambient property, this will be the default value.

Remarks

When a control is subclassed, there are certain messages that are normally sent to the parent control. Under normal conditions, these messages are actually reflected back to the sending control, so that the control can handle its own message. This message reflection can be handled by the container, which will reflect the messages back as events. The **MessageReflect** property tells if the container for the control does message reflection.

If the control is ever placed in a container that does not reflect messages, the operation of the control will be severely compromised; much of the operation of a control depends on reflected messages.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

Min Property (MSChart)

See Also Example Applies To

Returns a reference to a **Coor** object that specifies the starting corner of a rectangle.

Syntax

object.**Min**

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

Min Property

See Also Example Applies To

Sets or returns the minimum value of the scroll range for the **UpDown** control.

Syntax

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

The **Min** 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 value that specifies the minimum value, as described below. The setting for *value* can be a negative number.

Remarks

Pressing the down or left arrow normally causes the **UpDown** control to decrease the **Value** property. However, if the **Min** property is greater than the **Max** property, the **UpDown** control operates in the reverse direction.

Pressing the down or left arrow always causes the **Value** property to approach the **Min** value. Pressing the up or right arrow always causes the **Value** property to approach the **Max** value.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MinButton Property

See Also Example Applies To

Returns a value indicating whether a form has a Minimize button.

Syntax

object.**MinButton**

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

Return Values

The **MinButton** return values are:

Setting	Description
True	(Default) The form has a Minimize button.
False	The form doesn't have a Minimize button.

Remarks

A Minimize button enables users to minimize a form window to an icon. To display a Minimize button, you must also set the form's **BorderStyle** property to either 1 (Fixed Single), 2 (Sizable), or 3 (Fixed Double).

The settings you specify for the **MaxButton**, **MinButton**, **BorderStyle**, and **ControlBox** properties aren't reflected in the form's appearance until run time.

Note Minimizing a form to an icon at run time generates a Resize event. The **WindowState** property reflects the current state of the window. If you set the **WindowState** property to 2 (Maximized), the form is maximized independently of whatever settings are in effect for the **MaxButton** and **BorderStyle** properties.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MinHeight, MinWidth Properties

See Also Example Applies To

Returns or sets the minimum height or width of the Viewport at which scrollbars will appear on the container.

Syntax

object.**MinHeight** = *single*

object.**MinWidth** = *single*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
single	The height or width of a UserDocument at which scrollbars will appear on a container.

Remarks

The default values of the **MinHeight** and **MinWidth** properties are set by the **Height** and **Width** properties of the **UserDocument**.

The **MinWidth** and **MinHeight** have no effect if the **ScrollBars** property is set to **False**.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

Minimum Property

See Also Example Applies To

Returns or sets the lowest or beginning value on the chart value axis.

Syntax

object.**Minimum** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	Double. The lowest axis value.

Remarks

If this property is set, then the **ValueScale** object's **Auto** property is automatically set to **False**.

The **Maximum** property should be set before the **Minimum** property to avoid a chart display error.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Minor Property

See Also Example Applies To

Returns or sets the minor release number of the project. Read only at run time.

Syntax

object.**Minor**

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

Remarks

The value of the **Minor** property is in the range from 0 to 9999.

This property provides version information about the running application.

You can set this property at design time in the Minor box in the Make tab of the Project Properties dialog box.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

Minor Property

See Also Example Applies To Specifics

Returns a Long indicating the minor version number of the referenced type library. Read-only.

Remarks

The number returned by the **Minor** property corresponds to the minor version number stored in the type library to which you have set the reference.

© 2017 Microsoft

Visual Basic Extensibility Reference

Minor Property Example

The following example uses the **Minor** property to return the minor version number of the specified **Reference** object in a particular project.

```
Debug.Print Application.VBE.VBProjects(1).References(1).Minor
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

MinorDivision Property

See Also Example Applies To

Returns or sets the number of minor divisions displayed on the axis.

Syntax

object.**MinorDivision** [= *num*]

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

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
num	Integer. Number of minor divisions.

Remarks

If this property is set, then the **ValueScale** object's **Auto** property is automatically set to **False**.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

MinorPen Property

See Also Example Applies To

Returns a reference to a **Pen** object that describes the appearance of the minor axis grid lines.

Syntax

object.**MinorPen**

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

Minute Property

See Also Example Applies To

Returns or sets a value that specifies the current minute that is displayed.

Syntax

object.**Minute** [= *value*]

The **Minute** 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 the current minute that is displayed.

Remarks

The **Minute** property can be any integer from 0 to 59.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

MinWidth Property

See Also Example Applies To

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

Syntax

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

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

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

Remarks

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

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

© 2017 Microsoft

Visual Basic: Windows Controls

MinWidth Property Example

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

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

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MiscFlags Property

See Also Example Applies To

Returns or sets a value that determines access to one or more additional features of the **OLE** container control.

Syntax

object.**MiscFlags** [= *value*]

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

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Value	An integer or constant specifying access to an additional feature, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
VbOLEMiscFlagMemStorage	1	Causes the control to use memory to store the object while it's loaded.
VbOLEMiscFlagDisableInPlace	2	Overrides the control's default behavior of allowing in-place activation for objects that support it.

Remarks

The **vbOLEMiscFlagMemStorage** flag setting is faster than the object's default action, which is to store it on disk as a temporary file. This setting can, however, use a great deal of memory for objects whose data requires a lot of space, such as a bitmap for a paint program.

If an object supports in-place activation, you can use the **vbOLEMiscFlagDisableInPlace** setting to force the object to activate in a separate window.

To combine values, use the **Or** operator. For example, to combine both flags, you could use this code:

```
Ole1.MiscFlags = vbOLEMiscFlagMemStorage Or _ vbOLEMiscFlagDisableInPlace
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

MixedState Property

See Also Example Applies To

Returns or sets a value that determines if a **Button** object in a **Toolbar** control appears in an indeterminate state.

Syntax

object.**MixedState** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to a **Button** object.
boolean	A Boolean expression that determines if a **Button** shows the indeterminate state, as specified in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The **Button** object is in the indeterminate state and becomes dimmed.
False	The **Button** object is not in the indeterminate state and looks normal.

Remarks

The **MixedState** property is typically used when a selection contains a variety of attributes. For example, if you select text that contains both plain (normal) characters and bold characters, the **MixedState** property is used. The image displayed by the **Button** object could then be changed to indicate its state, which would differ from the Checked and Unchecked value returned by the **Value** property.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Multimedia MCI Control

Visual Studio 6.0

Mode Property (Multimedia MCI Control)

See Also Example Applies To

Returns the current mode of an open MCI device. This property is not available at design time and is read-only at run time.

Syntax

[*form.*]*MMControl*.**Mode**

Remarks

The following table lists the **Mode** property return values for the **Multimedia MCI** control.

Value	Setting/Device mode	Description
524	**mciModeNotOpen**	Device is not open.
525	**mciModeStop**	Device is stopped.
526	**mciModePlay**	Device is playing.
527	**mciModeRecord**	Device is recording.
528	**mciModeSeek**	Device is seeking.
529	**mciModePause**	Device is paused.
530	**mciModeReady**	Device is ready.

Data Type

Long

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

Mode Property

See Also Example Applies To Specifics

Returns a value containing the mode of the specified project. Read-only.

Return Values

The **Mode** property return values are:

Constant	Description
vbext_vm_Run	The specified project is in run mode.
vbext_vm_Break	The specified project is in break mode.
vbext_vm_Design	The specified project is in design mode.

© 2017 Microsoft

Visual Basic Extensibility Reference

Mode Property Example

The following example uses the **Mode** property to return the mode of the active project. The value returned is a predefined constant representing the project's mode.

```
Debug.Print Application.VBE.ActiveVBProject.Mode
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

Month Property

See Also Example Applies To

Returns or sets a value that specifies the current month.

Syntax

object.**Month** [= *number*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A constant or numeric expression that evaluates to an integer indicating the month, as shown in Settings.

Settings

The settings for *number* are:

Constant	Value	Description
mvwJanuary	1	January
mvwFebruary	2	February
mvwMarch	3	March
mvwApril	4	April
mvwMay	5	May
mvwJune	6	June
mvwJuly	7	July
mvwAugust	8	August
mvwSeptember	9	September

mvwOctober	10	October
mvwNovember	11	November
mvwDecember	12	December

Remarks

The **Month** property can be set to any integer from 1 to 12.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

MonthBackColor Property

See Also Example Applies To

Returns or sets a value that specifies the background color displayed within a month.

Syntax

object.**MonthBackColor** [= *color*]

The **MonthBackColor** 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 background color displayed within a month, 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 (VBRUN) object library in the Object Browser. The Windows operating system substitutes the user's choices as specified in the Control Panel settings.

Remarks

The **MonthBackColor** property can be used with the **TitleBackColor**, **TitleForeColor** and **TrailingForeColor** properties to customize the colors of the control.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0; the lower three 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 (VBRUN) object library in the Object Browser.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

MonthColumns, MonthRows Properties

See Also Example Applies To

Returns or sets a value that specifies the number of months to be displayed horizontally and vertically.

Syntax

object.**MonthColumns** [= *number*]

object.**MonthRows** [= *number*]

The **MonthColumns** and **MonthRows** property syntaxs have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression specifying the number of months.

Remarks

The **MonthColumns and MonthRows** properties give you the ability to display more than one month at a time.

The **MonthColumns** property allows you to specify the number of months that will be displayed horizontally. The **MonthRows** property allows you to specify the number of months that will be displayed vertically.

The control can display up to twelve months.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

MouseCol, MouseRow Properties

SeeAlso Example Applies To

Returns the current mouse position, in row and column coordinates.

Syntax

object.**MouseCol** [=*value*]
object.**MouseRow** [=*value*]

Syntax for the **MouseCol** and **MouseRow** properties has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	The row and column coordinates that specify the current mouse position.

Remarks

Use these properties programmatically to determine the mouse location. These properties are useful in displaying context-sensitive help for individual cells and testing whether the user has clicked on a fixed row or column.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MouseIcon Property

See Also Example Applies To

Returns or sets a custom mouse icon.

Syntax

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

object.**MouseIcon [**= *picture***]**

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

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

Remarks

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

The **MouseIcon** property provides your program with easy access to custom cursors of any size, with any desired hot spot location. Visual Basic does not load animated cursor (.ani) files, even though 32-bit versions of Windows support these cursors.

© 2017 Microsoft

Visual Basic Reference

MouseIcon Property Example

This example illustrates how the **MouseIcon** property sets a custom mouse icon. To try the example, create a **ListBox** control on a form, and then set the **MultiSelect** property to 1 or 2. At run time, select one or more items. Different icons will appear, depending on whether you selected a single item or multiple items.

```
Private Sub Form_Load ()
   ' Put some items in the ListBox.
   List1.AddItem "Selection 1"
   List1.AddItem "Selection 2"
   List1.AddItem "Selection 3"
   List1.AddItem "Selection 4"
   List1.AddItem "Selection 5"
End Sub

Private Sub List1_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
   ' Set the custom mouse icon for multiple items.
   If List1.SelCount > 1 Then
      List1.MouseIcon = LoadPicture("ICONS\COMPUTER\MOUSE04.ICO")
      List1.MousePointer = 99
   Else   ' Set the custom mouse icon for a single item.
      List1.MouseIcon = LoadPicture("ICONS\COMPUTER\MOUSE02.ICO")
      List1.MousePointer = 99
   End If
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MouseIcon Property (ActiveX Controls)

See Also Example Applies To

Returns or sets a custom mouse icon.

Syntax

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

object.**MouseIcon [**= *picture***]**

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

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

Remarks

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

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MousePointer Property

See Also Example Applies To

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

Syntax

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

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

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

Settings

The settings for *value* are:

Constant	Value	Description
vbDefault	0	(Default) Shape determined by the object.
vbArrow	1	Arrow.
vbCrosshair	2	Cross (crosshair pointer).
vbIbeam	3	I beam.
vbIconPointer	4	Icon (small square within a square).
vbSizePointer	5	Size (four-pointed arrow pointing north, south, east, and west).
vbSizeNESW	6	Size NE SW (double arrow pointing northeast and southwest).
vbSizeNS	7	Size N S (double arrow pointing north and south).

vbSizeNWSE	8	Size NW SE (double arrow pointing northwest and southeast).
vbSizeWE	9	Size W E (double arrow pointing west and east).
vbUpArrow	10	Up Arrow.
vbHourglass	11	Hourglass (wait).
vbNoDrop	12	No Drop.
vbArrowHourglass	13	Arrow and hourglass.
vbArrowQuestion	14	Arrow and question mark.
vbSizeAll	15	Size all.
vbCustom	99	Custom icon specified by the **MouseIcon** property.

Remarks

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

Note If your application calls DoEvents, the **MousePointer** property may temporarily change when over an ActiveX component.

© 2017 Microsoft

Visual Basic Reference

MousePointer Property Example

This example changes the mouse pointer to an hourglass while circles are drawn across the screen and then changes the hourglass back to a pointer at the end of the procedure. To try this example, paste the code into the Declarations section of a form. Press F5 to run the program, and then click the form.

```
Private Sub Form_Click ()
   Dim I    ' Declare variable.
   ' Change mouse pointer to hourglass.
   Screen.MousePointer = vbHourglass
 ' Set random color and draw circles on form.
   For I = 0 To ScaleWidth Step 50
      ForeColor = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
      Circle (I, ScaleHeight * Rnd), 400
   Next
   ' Return mouse pointer to normal.   Screen.MousePointer = vbDefault
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MousePointer Property (ActiveX Controls)

See Also Example Applies To

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

Syntax

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

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

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

Settings

The settings for *value* are:

Constant	Value	Description
vbDefault	0	(Default) Shape determined by the object.
VbArrow	1	Arrow.
VbCrosshair	2	Cross (crosshair pointer).
VbIbeam	3	I beam.
VbIconPointer	4	Icon (small square within a square).
VbSizePointer	5	Size (four-pointed arrow pointing north, south, east, and west).
VbSizeNESW	6	Size NE SW (double arrow pointing northeast and southwest).
VbSizeNS	7	Size N S (double arrow pointing north and south).

VbSizeNWSE	8	Size NW SE (double arrow pointing northwest and southeast).
VbSizeWE	9	Size W E (double arrow pointing west and east).
VbUpArrow	10	Up Arrow.
VbHourglass	11	Hourglass (wait).
VbNoDrop	12	No Drop.
VbArrowHourglass	13	Arrow and hourglass. (Only available in 32-bit Visual Basic.)
vbArrowQuestion	14	Arrow and question mark. (Only available in 32-bit Visual Basic.)
vbSizeAll	15	Size all. (Only available in 32-bit Visual Basic.)
vbCustom	99	Custom icon specified by the **MouseIcon** property.

Remarks

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

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

MousePointer Property (MSChart)

See Also Example Applies To

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

Syntax

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

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	Integer. A VtMousePointer constant that specifies the type of mouse pointer displayed.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Moveable Property

See Also Example Applies To

Returns or sets a value which specifies if the object can be moved.

Syntax

object.**Moveable** = *boolean*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies if the object can be moved.

Settings

The settings for *boolean* are:

Constant	Value	Description
True	-1	The object can be moved.
False	0	The object cannot be moved.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

MsgConversationID Property

See Also Example Applies To

Specifies the conversation thread identification value for the currently indexed message. It is read-only unless **MsgIndex** is set to 1.

Syntax

```
object.MsgConversationID [ = value ]
```

The **MsgConversationID** 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 specifying the conversation identification value.

Remarks

A conversation thread is used to identify a set of messages beginning with the original message and including all the subsequent replies. Identical conversation IDs indicate that the messages are part of the same thread. New messages are assigned an ID by the message system. The value of **MsgConversationID** depends on the currently indexed message, as selected by the **MsgIndex** property.

Data Type

String

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

MsgCount Property

See Also Example Applies To

Returns the total number of messages present in the message set during the current messaging session. This property is not available at design time, and is read-only at run time.

Syntax

object.**MsgCount**

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

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

Remarks

This property is used to get a current count of the messages in the message set. The default value is 0. This property is reset each time a fetch is performed.

Data Type

Long

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

MsgDateReceived Property

See Also Example Applies To

Returns the date on which the currently indexed message was received. This property is not available at design time and is read-only at run time.

Syntax

object.**MsgDateReceived**

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

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

Remarks

The format for this property is YYYY/MM/DD HH:MM. Hours are measured on a standard 24-hour base. The value of **MsgDateReceived** is set by the message system and depends on the currently indexed message, as selected by the **MsgIndex** property.

Data Type

String

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

MsgID Property

See Also Example Applies To

Returns the string identifier of the currently indexed message. This property is not available at design time and is read-only at run time.

Syntax

object.**MsgID**

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

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

Remarks

The message-identifier string is a system-specific, 64-character string used to uniquely identify a message. The value of **MsgID** depends on the currently indexed message, as selected by the **MsgIndex** property.

Data Type

String

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

MsgIndex Property

See Also Example Applies To

Specifies the index number of the currently indexed message. This property is not available at design time.

Syntax

object.**MsgIndex** [= *value*]

The **MsgIndex** property syntax these parts:

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Value	A long expression specifying the index number of the currently indexed message.

Remarks

The **MsgIndex** property determines the values of all the other message-related properties of the **MAPI Messages** control. The index number can range from -1 to **MsgCount** -1.

Note Changing the **MsgIndex** property also changes the entire set of attachments and recipients.

The message identified by the **MsgIndex** property is called the *currently indexed* message. When this index is changed, all of the other message properties change to reflect the characteristics of the indexed message. A value of -1 signifies a message being built in the compose buffer in other words, an outgoing message.

Data Type

Long

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

MsgNoteText Property

See Also Example Applies To

Specifies the text body of the message. This property is not available at design time. It is read-only unless **MsgIndex** is set to 1.

Syntax

object.**MsgNoteText** [= *value*]

The **MsgNoteText** 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 specifying message text.

Remarks

This property consists of the entire textual portion of the message body (minus any attachments). An empty string indicates no text.

For inbound messages, each paragraph is terminated with a carriage return-line feed pair (0x0d0a). For outbound messages, paragraphs can be delimited with a carriage return (0x0d), line feed (0x0a), or a carriage return-line feed pair (0x0d0a). The value of **MsgNoteText** depends on the currently indexed message, as selected by the **MsgIndex** property.

Data Type

String

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

MsgOrigAddress Property

See Also Example Applies To

Returns the mail address of the originator of the currently indexed message. This property is not available at design time and is read-only at run time. The messaging system sets this property for you when sending a message.

Syntax

object.**MsgOrigAddress**

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

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

Remarks

The value of **MsgOrigAddress** depends on the currently indexed message as selected by the **MsgIndex** property. The value is null in the compose buffer.

Data Type

String

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

MsgOrigDisplayName Property

See Also Example Applies To

Returns the originator's name for the currently indexed message. This property is not available at design time and is read-only at run time. The messaging system sets this property for you.

Syntax

object.**MsgOrigDisplayName**

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

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

Remarks

The name in this property is the originator's name, as displayed in the message header. The value of **MsgOrigDisplayName** depends on the currently indexed message, as selected by the **MsgIndex** property. The value is null in the compose buffer.

Data Type

String

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

MsgRead Property

See Also Example Applies To

Returns a boolean expression indicating whether the message has already been read. This property is not available at design time and is read-only at run time.

Syntax

object.**MsgRead**

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

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

Settings

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

Setting	Description
True	The currently indexed message has already been read by the user.
False	(Default) The message remains unread.

Remarks

The value of **MsgRead** depends on the currently indexed message, as selected by the **MsgIndex** property. The message is marked as read when the note text or any of the attachment information is accessed. However, accessing header information does not mark the message as read.

Data Type

Boolean

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

MsgReceiptRequested Property

See Also Example Applies To

Specifies whether a return receipt is requested for the currently indexed message. This property is not available at design time.

Syntax

object.**MsgReceiptRequested** [= *value*]

The **MsgReceiptRequested** 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 a return receipt is requested, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	A receipt notification is returned to the sender when the recipient opens the message.
False	(Default) No return receipt is generated.

Remarks

The value of **MsgReceiptRequested** depends on the currently indexed message, as selected by the **MsgIndex** property.

Data Type

Boolean

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

MsgSent Property

See Also Example Applies To

Specifies whether the currently indexed message has already been sent to the mail server for distribution. This property is not available at design time and is read-only at run time. The messaging system sets this property for you when sending a message.

Syntax

object.**MsgSent**

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

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

Settings

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

Setting	Description
True	The currently indexed message has already been submitted to the mail server as an outgoing message.
False	The currently indexed message has not yet been delivered to the server.

Remarks

The value of **MsgSent** depends on the currently indexed message, as selected by the **MsgIndex** property.

Data Type

Boolean

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

MsgSubject Property

See Also Example Applies To

Specifies the subject line for the currently indexed message as displayed in the message header. This property is not available at design time. It is read-only unless **MsgIndex** is set to -1.

Syntax

```
object.MsgSubject [ = value ]
```

The **MsgSubject** 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 specifying the subject line.

Remarks

The value of **MsgSubject** depends on the currently indexed message, as selected by the **MsgIndex** property. **MsgSubject** is limited to 64 characters, including the Null character.

Data Type

String

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

MsgType Property

See Also Example Applies To

Specifies the type of the currently indexed message. This property is not available at design time. It is read-only unless **MsgIndex** is set to -1.

Syntax

object.**MsgType** [= *value*]

The **MsgType** 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 specifying the type of message.

Remarks

The **MsgType** property is for use by applications other than interpersonal mail (IPM message type). Not all mail systems support message types that are not IPM and may not provide (or may ignore) this parameter.

A null or empty string indicates an IPM message type. The value of **MsgType** depends on the currently indexed message, as selected by the **MsgIndex** property. This property is not meant for use as a filter to isolate messages by sender, receipt time, and other categories.

Data Type

String

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MultiLine Property

See Also Example Applies To

Returns or sets a value indicating whether a **TextBox** control can accept and display multiple lines of text. Read only at run time.

Syntax

object.**MultiLine**

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

Settings

The **MultiLine** property settings are:

Setting	Description
True	Allows multiple lines of text.
False	(Default) Ignores carriage returns and restricts data to a single line.

Remarks

A multiple-line **TextBox** control wraps text as the user types text extending beyond the text box.

You can also add scroll bars to larger **TextBox** controls using the **ScrollBars** property. If no horizontal scroll bar is specified, the text in a multiple-line **TextBox** automatically wraps.

Note On a form with no default button, pressing ENTER in a multiple-line **TextBox** control moves the focus to the next line. If a default button exists, you must press CTRL+ENTER to move to the next line.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: RichTextBox Control

Visual Studio 6.0

MultiLine Property (RichTextBox Control)

See Also Example Applies To

Returns or sets a value indicating whether a **RichTextBox** control can accept and display multiple lines of text. Read-only at run time.

Syntax

object.**MultiLine**

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

Settings

The **MultiLine** property settings are:

Setting	Description
True	Allows multiple lines of text.
False	(Default) Ignores carriage returns and restricts data to a single line.

Remarks

A multiple-line **RichTextBox** control wraps text as the user types text extending beyond the text box.

You can also add scroll bars to a larger **RichTextBox** control using the **ScrollBars** property. If no **HScrollBar** control (horizontal scroll bar) is specified, the text in a multiple-line **RichTextBox** automatically wraps.

Note On a form with no default button, pressing ENTER in a multiple-line **RichTextBox** control moves the focus to the next line. If a default button exists, you must press CTRL+ENTER to move to the next line.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

MultiRow Property

See Also Example Applies To

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

Syntax

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

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

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

Settings

The settings for *boolean* are:

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

Remarks

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

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

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

MultiSelect Property

See Also Example Applies To

Returns or sets a value indicating whether a user can make multiple selections in a **FileListBox** or **ListBox** control and how the multiple selections can be made. Read only at run time.

Syntax

object.**MultiSelect**

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

Settings

The **MultiSelect** property settings are:

Setting	Description
0	(Default) Multiple selection isn't allowed.
1	Simple multiple selection. A mouse click or pressing the SPACEBAR selects or deselects an item in the list. (Arrow keys move the focus.)
2	Extended multiple selection. Pressing SHIFT and clicking the mouse or pressing SHIFT and one of the arrow keys (UP ARROW, DOWN ARROW, LEFT ARROW, and RIGHT ARROW) extends the selection from the previously selected item to the current item. Pressing CTRL and clicking the mouse selects or deselects an item in the list.

© 2017 Microsoft

Visual Basic Reference

MultiSelect Property Example

This example fills a **ListBox** control with the names of your screen fonts and illustrates how the **MultiSelect** property affects the behavior of a **ListBox**. To try this example, create two **ListBox** controls and a **CommandButton** control on a form. In the first **ListBox**, set the **MultiSelect** property to 1 or 2. At run time, select several items in the first **ListBox**, and then click the **CommandButton**. All selected items are displayed in the second **ListBox**. Run the example several times with different settings of the **MultiSelect** property. Paste the code into the Declarations section, and then press F5 to run the program.

```
Private Sub Form_Load ()
   Dim I   ' Declare variable.
   ' Fill the list box with screen font names.
   For I = 0 To Screen.FontCount - 1
      List1.AddItem Screen.Fonts(I)
   Next I
End Sub

Private Sub Command1_Click ()
   Dim I   ' Declare variable.
   ' Clear all items from the list.
   List2.Clear
   ' If an item is selected, add it to List2.
   For I = 0 To List1.ListCount - 1
      If List1.Selected(I) Then
         List2.AddItem List1.List(I)
      End If
   Next I
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

MultiSelect Property (ListView, TabStrip Controls)

See Also Example Applies To

Returns or sets a value indicating whether a user can select multiple objects or items.

Syntax

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

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

Part	Description
object	An object expression that evaluates to an item in the Applies To list
boolean	A value specifying the type of selection, as described in Settings.

Settings

The settings for *boolean* are:

Constant	Description
False	(Default) Selecting multiple objects or items isn't allowed.
True	Multiple selection allowed.

Remarks

For the **ListView** control: pressing SHIFT and clicking the mouse or pressing SHIFT and one of the arrow keys (UP ARROW, DOWN ARROW, LEFT ARROW, and RIGHT ARROW) extends the selection from the previously selected **ListItem** to the current **ListItem**. Pressing CTRL and clicking the mouse selects or deselects a **ListItem** in the list.

For the **TabStrip** control: **Tab** objects can only be selected when the **Style** property is set to **TabButton**.

© 2017 Microsoft

Visual Basic: Windows Controls

Ghosted, MultiSelect Properties Example

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

Note The example will not run unless you add a reference to the Microsoft DAO 3.51 Object Library by using the References command on the Tools menu. Run the example, select a **MultiSelect** option by clicking an **OptionButton**, click on items to select them and click the **CommandButton** to ghost them.

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

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

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

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

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

```
    ' Set the recordset to the Publishers table.
    Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

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

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

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

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

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

       myRs.MoveNext  ' Move to next record.
    Wend

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

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

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

MultiSelect Property (MonthView Control)

See Also Example Applies To

Returns or sets a value that determines if multiple dates can be selected at once.

Syntax

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

The **MultiSelect** 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 selection behavior exhibited by the control.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) The user is allowed to select multiple days.
False	The user is not allowed to select multiple days.

Remarks

By default, the control allows the user to select a range of dates. The default maximum range is one week (7 days). You can change the maximum selectable range by setting the **MaxSelCount** property. The **Value** property will be in this range, indicating which date has focus.

© 2017 Microsoft