Visual Basic: MAPI Controls

Visual Studio 6.0

Save Method

See Also Example Applies To

Saves the message currently in the compose buffer (with **MsgIndex** = -1).

Syntax

object.**Save**

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

Visual Basic Extensibility Reference

Visual Studio 6.0

SaveAs Method

See Also Example Applies To

Saves a component or project to a given location using a new filename.

Syntax

object.**SaveAs (***newfilename* **As String)**

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
newfilename	Required. A string expression specifying the new filename for the component to be saved.

Remarks

If a new path name is given, it is used. Otherwise, the old path name is used. If the new filename is invalid or refers to a read-only file, an error occurs.

When a form is saved, *newfilename* specifies the new name of the form file itself. The .Frx file, if applicable, is saved automatically with an .Frx extension.

Note Successfully invoking this method causes the associated events from the **FileControl** object to be invoked.

Visual Basic: RichTextBox Control

Visual Studio 6.0

SaveFile Method

See Also Example Applies To

Saves the contents of a **RichTextBox** control to a file.

Syntax

object.**SaveFile**(*pathname*, *filetype*)

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
pathname	Required. A string expression defining the path and filename of the file to receive the contents of the control.
filetype	Optional. An integer or constant that specifies the type of file loaded, as described in Settings.

Settings

The settings for *filetype* are:

Constant	Value	Description
rtfRTF	0	(Default) RTF. The **RichTextBox** control saves its contents as an .rtf file.
rtfText	1	Text. The **RichTextBox** control saves its contents as a text file.

Remarks

You can also use the **Write** function in Visual Basic and the **TextRTF** and **SelRTF** properties of the **RichTextBox** control to write .rtf files. For example, you can save the highlighted contents of a **RichTextBox** control to an .rtf file as follows:

```
Open "mytext.rtf" For Output As 1

Print #1, RichTextBox1.SelRTF
```

Visual Basic: RichTextBox Control

SaveFile Method Example

This example displays a dialog box to choose an .rtf file to which you will save the contents of a **RichTextBox** control. To try this example, put a **RichTextBox** control, a **CommandButton** control, and a **CommonDialog** control on a form. Paste this code into the Click event of the **CommandButton** control. Then run the example.

```
Private Sub Command1_Click()
   CommonDialog1.ShowSave
   RichTextBox1.SaveFile CommonDialog1.Filename, rtfRTF
End Sub
```

Visual Basic Reference

Visual Studio 6.0

SaveToFile Method

See Also Example Applies To

Saves an object to a data file. Doesn't support named arguments.

Syntax

object.**SaveToFile** *filenumber*

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

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Filenumber	Required. A numeric expression specifying the file number used when saving an object. This number must correspond to an open, binary file.

Remarks

Use this method to save ActiveX components. To save an ActiveX component in the OLE version 1.0 format, use the **SaveToOle1File** method instead.

If the object is linked (**OLEType** = **vbOLELinked**, 0), only the link information and an image of the data is saved to the specified file. The object's data is maintained by the application that created the object. If the object is embedded (**OLEType** = **vbOLEEmbedded**, 1), the object's data is maintained by the **OLE** container control and can be saved by your Visual Basic application.

You can load an object saved to a data file with the **ReadFromFile** method.

Visual Basic: Windows Controls

Visual Studio 6.0

SaveToolbar Method

See Also Example Applies To

At run time, saves the state of a toolbar, created with the **Toolbar** control, in the registry.

Syntax

object.**SaveToolbar(***key* **As String,** *subkey* **As String**, *value* **As String)**

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

Part	Description
object	Required. An object expression that evaluates to a **Toolbar** control.
key	Required. A string expression specifying the key in the registry where the method stores the **Toolbar** information.
subkey	Required. A string expression that specifies a location in the registry under the *key* parameter.
value	Required. The **Toolbar** information to be stored in the *subkey*.

Remarks

To customize the **Toolbar** control at run time, use the **Customize** method in code or if the **AllowCustomize** property is **True**, the user can customize it by double clicking the control.

If the *key*, *subkey, or value* you specify doesn't exist in the registry, it is created.

To save more than one version of the toolbar, you can change the *subkey* or *value* parameter. This causes the toolbar to write to a different part of the registry. The following code saves two different states of a toolbar after it has been customized.

```
' Save settings for User1
Toolbar1.SaveToolbar "AppName", "User1", "Toolbar1"

' Save settings for User2
Toolbar1.SaveToolbar "AppName", "User2", "Toolbar1"
```

Since the Change event for the **Toolbar** control occurs after the toolbar has been customized, in most cases the above code can be placed in the Change event for the toolbar.

Visual Basic Reference

Visual Studio 6.0

SaveToOle1File Method

See Also Example Applies To

Saves an object in the OLE version 1.0 file format. Doesn't support named arguments.

Syntax

object.**SaveToOle1File** *filenumber*

The **SaveToOle1File** method syntax has these parts:

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Filenumber	Required. A numeric expression specifying the file number used when saving or loading an object. This number must correspond to an open, binary file.

Remarks

If the object is linked (**OLEType** = **vbOLELinked**, 0), only the link information and an image of the data is saved to the specified file. The object's data is maintained by the application that created the object. If the object is embedded (**OLEType** = **vbOLEEmbedded**, 1), the object's data is maintained by the **OLE** container control and can be saved by your Visual Basic application.

If you want to save the object in the current ActiveX component format, use the **SaveToFile** method instead.

Visual Basic Reference

Visual Studio 6.0

Scale Method

See Also Example Applies To

Defines the coordinate system for a **Form**, **PictureBox**, or **Printer**. Doesn't support named arguments.

Syntax

object.**Scale (***x1*, *y1***) - (***x2*, *y2***)**

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

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** object with the focus is assumed to be *object*.
x1, *y1*	Optional. Single-precision values indicating the horizontal (x-axis) and vertical (y-axis) coordinates that define the upper-left corner of *object*. Parentheses must enclose the values. If omitted, the second set of coordinates must also be omitted.
x2, *y2*	Optional. Single-precision values indicating the horizontal and vertical coordinates that define the lower-right corner of *object*. Parentheses must enclose the values. If omitted, the first set of coordinates must also be omitted.

Remarks

The **Scale** method enables you to reset the coordinate system to any scale you choose. **Scale** affects the coordinate system for both run-time graphics statements and the placement of controls.

If you use **Scale** with no arguments (both sets of coordinates omitted), it resets the coordinate system to twips.

Visual Basic Reference

Scale Method Example

This example uses the **Scale** method to set up a custom coordinate system so that a bar chart can be drawn on a form. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim I, OldFontSize   ' Declare variables.
   Width = 8640: Height = 5760   ' Set form size in twips.
   Move 100,100   ' Move form origin.
   AutoRedraw = -1   ' Turn on AutoRedraw.
   OldFontSize = FontSize   ' Save old font size.
   BackColor = QBColor(7)   ' Set background to gray.
   Scale (0, 110)-(130, 0)   ' Set custom coordinate system.
   For I = 100 To 10 Step -10
      Line (0, I)-(2, I)   ' Draw scale marks every 10 units.
      CurrentY = CurrentY + 1.5   ' Move cursor position.
      Print I   ' Print scale mark value on left.
      Line (ScaleWidth - 2, I)-(ScaleWidth, I)
      CurrentY = CurrentY + 1.5   ' Move cursor position.
      CurrentX = ScaleWidth - 9
      Print I   ' Print scale mark value on right.
   Next I
   ' Draw bar chart.
   Line (10, 0)-(20, 45), RGB(0, 0, 255), BF   ' First blue bar.
   Line (20, 0)-(30, 55), RGB(255, 0, 0), BF   ' First red bar.
   Line (40, 0)-(50, 40), RGB(0, 0, 255), BF
   Line (50, 0)-(60, 25), RGB(255, 0, 0), BF
   Line (70, 0)-(80, 35), RGB(0, 0, 255), BF
   Line (80, 0)-(90, 60), RGB(255, 0, 0), BF
   Line (100, 0)-(110, 75), RGB(0, 0, 255), BF
   Line (110, 0)-(120, 90), RGB(255, 0, 0), BF
   CurrentX = 18: CurrentY = 100   ' Move cursor position.
   FontSize = 14   ' Enlarge font for title.
   Print "Widget Quarterly Sales"   ' Print title.
   FontSize = OldFontSize   ' Restore font size.
   CurrentX = 27: CurrentY = 93   ' Move cursor position.
   Print "Planned Vs. Actual"   ' Print subtitle.
   Line (29, 86)-(34, 88), RGB(0, 0, 255), BF   ' Print legend.
   Line (43, 86)-(49, 88), RGB(255, 0, 0), BF
End Sub
```

Visual Basic Reference

Visual Studio 6.0

ScaleX, ScaleY Methods

See Also Example Applies To

Converts the value for the width or height of a **Form**, **PictureBox**, or **Printer** from one of the **ScaleMode** property's unit of measure to another. Doesn't support named arguments.

Syntax

object.**ScaleX (***width*, *fromscale*, *toscale***)**

object.**ScaleY (***height*, *fromscale*, *toscale***)**

The **ScaleX** and **ScaleY** method syntaxes have these parts:

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** object with the focus is assumed to be *object*.
width	Required. Specifies, for *object*, the number of units of measure to be converted.
height	Required. Specifies, for *object*, the number of units of measure to be converted.
fromscale	Optional. A constant or value specifying the coordinate system from which *width* or *height* of *object* is to be converted, as described in Settings. The possible values of *fromscale* are the same as for the **ScaleMode** property, plus the new value of HiMetric.
toscale	Optional. A constant or value specifying the coordinate system to which *width* or *height* of *object* is to be converted, as described in Settings. The possible values of *toscale* are the same as for the **ScaleMode** property, plus the new value of HiMetric.

Settings

The settings for *fromscale* and *toscale* are:

Constant	Value	Description
vbUser	0	User-defined: indicates that the width or height of *object* is set to a custom value.
vbTwips	1	Twip (1440 twips per logical inch; 567 twips per logical centimeter).

vbPoints	2	Point (72 points per logical inch).
vbPixels	3	Pixel (smallest unit of monitor or printer resolution).
vbCharacters	4	Character (horizontal = 120 twips per unit; vertical = 240 twips per unit).
vbInches	5	Inch.
vbMillimeters	6	Millimeter.
vbCentimeters	7	Centimeter.
vbHimetric	8	HiMetric. If *fromscale* is omitted, HiMetric is assumed as the default.
vbContainerPosition	9	Determines control's position.
vbContainerSize	10	Determines control's size.

Remarks

The **ScaleX** and **ScaleY** methods take a value (*width* or *height*), with its unit of measure specified by *fromscale*, and convert it to the corresponding value for the unit of measure specified by *toscale*.

You can also use **ScaleX** and **ScaleY** with the **PaintPicture** method.

Visual Basic: DataGrid Control

Visual Studio 6.0

Scroll Method

See Also Example Applies To

Scrolls the **DataGrid** control horizontally and vertically in a single operation. Doesn't support named arguments.

Syntax

object.**Scroll** *colvalue, rowvalue*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
colvalue	Required. A long numeric expression that specifies a column in the control.
rowvalue	Required. A long numeric expression that specifies a row in the control.

Remarks

Positive values scroll right and down. Negative values scroll left and up. Values that are out of range don't cause an error the **DataGrid** control scrolls to the maximum degree possible. The same effect can be achieved by setting the **FirstRow** and **LeftCol** properties, but these must be set independently, causing two separate Paint events.

Visual Basic: DataGrid Control

Scroll Method Example

This example creates two buttons that enable you to scroll diagonally, one to move down and to the right and the other to move up and to the left.

```
Sub ScrollDownRight_Click
   ' Scroll down and to the right.
   DataGrid1.Scroll DataGrid1.VisibleCols, DataGrid1.VisibleRows
End Sub

Sub ScrollUpLeft_Click
   ' Scroll up and to the left.
   DataGrid1.Scroll -DataGrid1.VisibleCols, -DataGrid1.VisibleRows
End Sub
```

Visual Basic: MSChart Control

Select Method

See Also Example Applies To

Selects the specified chart element.

Syntax

object.**Select**

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

Visual Basic Reference

Visual Studio 6.0

SelectAll Method

See Also Example Applies To

Selects all of the controls contained on a form.

Syntax

object.**SelectAll**

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

Visual Studio 6.0

Visual Basic: MSChart Control

SelectPart Method

See Also Example Applies To

Selects the specified chart part.

Syntax

object.**SelectPart** (*part*, *index1*, *index2*, *index3*, *index4*)

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
part	Integer. Specifies the chart part. Valid constants are VtChPartType.
index1	Integer. If *part* refers to a series or a data point, this argument specifies which series. Series are numbered in the order their corresponding columns appear in the data grid from left to right, beginning with 1. If *part* refers to an axis or axis label, this argument identifies the axis type with a VtChAxisId constant.
index2	Integer. If *part* refers to a data point, this argument specifies which data point in the series is identified by index1. Data points are numbered in the order their corresponding rows appear in the data grid from top to bottom, beginning with 1. If *part* refers to an axis, axis title, or axis label, this argument refers to the axis index which is currently not used. In this case, the only valid value for this argument is 1.
index3	Integer. If *part* refers to an axis label, this argument refers to the level of the label. Axis label levels are numbered from the axis out, beginning with 1. If *part* is not an axis label, the argument is unused.
index4	Integer. This argument is unused at this time.

Visual Basic: RichTextBox Control

Visual Studio 6.0

SelPrint Method

See Also Example Applies To

Sends formatted text in a **RichTextBox** control to a device for printing.

Syntax

object.**SelPrint(***hdc [,vStartDoc]***)**

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
hdc	The device context of the device you plan to use to print the contents of the control.
vStartDoc	Boolean. Specifies the behavior of the control regarding startdoc and enddoc printer control operations, as shown in settings.

Settings

The settings for *vStartDoc* are:

Constant	Value	Description
True	-1	(Default) The control retains its original behavior and sends startdoc and enddoc commands to the printer.
False	0	The control doesn't send startdoc and enddoc commands, but sends only startpage and endpage commands to the printer.

Remarks

If text is selected in the **RichTextBox** control, the **SelPrint** method sends only the selected text to the target device. If no text is selected, the entire contents of the **RichTextBox** are sent to the target device.

The **SelPrint** method does not print text from the **RichTextBox** control. Rather, it sends a copy of formatted text to a device which can print the text. For example, you can send the text to the **Printer** object using code as follows:

```
RichTextBox1.SelPrint(Printer.hDC)
```

Notice that the **hDC** property of the **Printer** object is used to specify the device context argument of the **SelPrint** method.

Note If you use the **Printer** object as the destination of the text from the **RichTextBox** control, you must first initialize the device context of the **Printer** object by printing something like a zero-length string.

The *vStartDoc* argument remedies situations when printers do not print with the default behavior. When the **SelPrint** method is invoked, both Visual Basic and the **RichTextBox** control send *startdoc* and *enddoc* commands to the printer resulting in a nested pair of startdoc/enddoc commands. Some printers respond only to the first pair of commands and thereby become disabled when the **RichTextbox** control sends the second pair. In that case, setting the *vStartDoc* argument to **False** prevents the second pair of commands from being sent.

Visual Basic: RichTextBox Control

SelPrint Method Example

This example prints the formatted text in a **RichTextBox** control. To try this example, put a **RichTextBox** control, a **CommonDialog** control, and a **CommandButton** control on a form. Paste this code into the Click event of the **CommandButton** control. Then run the example.

```
Private Sub Command1_Click()
   CommonDialog1.Flags = cdlPDReturnDC + cdlPDNoPageNums
   If RichTextBox1.SelLength = 0 Then
      CommonDialog1.Flags = CommonDialog1.Flags + cdlPDAllPages
   Else
      CommonDialog1.Flags = CommonDialog1.Flags + cdlPDSelection
   End If
   CommonDialog1.ShowPrinter
   Printer.Print ""
   RichTextBox1.SelPrint CommonDialog1.hDC
End Sub
```

Visual Basic: MAPI Controls

Visual Studio 6.0

Send Method

See Also Example Applies To

Sends a message.

Syntax

object.**Send** [*value*]

The **Send** method 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 to show a dialog box, as described in settings.

Settings

The settings for *value* are:

Setting	Description
True	Sends a message inside a dialog box. Prompts the user for the various components of the message and submits the message to the mail server for delivery.
	All message properties associated with the message being built in the compose buffer form the basis for the message dialog box. However, changes made in the dialog box do not alter information in the compose buffer.
False	(Default) Submits the outgoing message to the mail server without displaying a dialog box. An error occurs if you attempt to send a message with no recipients or with missing attachment path names.

Visual Basic: Winsock Control

Visual Studio 6.0

SendData Method

See Also Example Applies To

Sends data to a remote computer.

Return Value

Void

Syntax

object.**SendData** *data*

The **SendData** method syntax has these parts

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
data	Data to be sent. For binary data, byte array should be used.

Remarks

When a UNICODE string is passed in, it is converted to an ANSI string before being sent out on the network.

Visual Studio 6.0

Visual Basic: MSChart Control

Set Method (Coor, LCoor)

See Also Example Applies To

Sets the *x* and *y* coordinate values for a chart.

Syntax

object.**Set** (*x*,*y*)

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
x	Single. (Long for **LCoor** object.) Identifies the *x* value of the coordinate.
y	Single. (Long for **LCoor** object.) Identifies the *y* value of the coordinate.

Visual Studio 6.0

Visual Basic: MSChart Control

Set Method (LightSource)

See Also Example Applies To

Sets the *x*, *y*, and *z* coordinates and the intensity for the **LightSource** object location.

Syntax

object.**Set** (*x*,*y*,*z*, *intensity*)

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
x, y, z	Integers. Indicate the light source location.
intensity	Single. Indicate the light source intensity.

Visual Studio 6.0

Visual Basic: MSChart Control

Set Method (View3D)

See Also Example Applies To

Sets the rotation and degree of elevation for a three-dimensional chart.

Syntax

object.**Set** (*rotation*, *elevation*)

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
rotation	Single. The degree of rotation. Rotation can range from 0 to 360 degrees. By default, degrees are used to measure rotation. However, these settings use the current settings for the **AngleUnit** property. The other options are: Grads and Radians.
elevation	Single. The degree of elevation. Elevation can be any number from 0 to 90 degrees. If you set the elevation to 90 degrees, you look directly down onto the top of the chart. If you set the elevation to 0, you look directly at the side of the chart. The default elevation is 30 degrees. By default, degrees are used to measure elevation. However, these settings use the current settings for the **AngleUnits** property. The other options are: Grads and Radians.

Visual Studio 6.0

Visual Basic: MSChart Control

Set Method (VtColor)

See Also Example Applies To

Sets the red, green and blue values of the **VtColor** object.

Syntax

object.**Set** (*red*,*green*,*blue*)

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
red, *green*, *blue*	Integer. The values for the red, green and blue components of color.

Remarks

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

Visual Studio 6.0

Visual Basic: MSChart Control

Set Method (Weighting)

See Also Example Applies To

Sets the basis and style of the **Weighting** object.

Syntax

object.**Set** (*basis*, *style*)

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
basis	A VtChPieWeightBasis constant that identifies the weighting type.
style	A VtChPieWeightStyle constant that identifies the weighting factor method.

Visual Basic Reference

Visual Studio 6.0

SetAutoServerSettings Method

See Also Example Applies To

Sets the Remote Automation registry values to meet ActiveX and Remote Automation requirements, including configuration settings for remote server access.

Syntax

object. **SetAutoServerSettings**(*remote*, [*progid*], [*clsid*], [*servername*], [*protocol*], [*authentication*])

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
remote	Required. Boolean. **True** if the server is remote, **False** if local.
progid	Optional. A variant expression specifying the ProgID for the server.
clsid	Optional. A variant expression specifying the CLSID for the server.
servername	Optional. A variant expression specifying the name of the server machine.
protocol	Optional. A variant expression specifying the RPC name of the protocol to be used.
authentication	Optional. A variant expression specifying the RPC authentication level.

Return Values

The **SetAutoServerSettings** method returns the following error codes:

Value	Description
0	No error.
1	Unknown run time error occurred.
2	No protocol was specified.

3	No server machine name was specified.
4	An error occurred reading from the registry.
5	An error occurred writing to the registry.
6	Both the ProgID and CLSID parameters were missing.
7	There is no local server (either in-process or cross-process, 16-bit or 32-bit).
8	There was an error looking for the Proxy DLLs, check that they were installed properly.

Remarks

The **SetAutoServerSettings** method takes either a CLSID or a ProgID and sets the registry information to local or remote depending on the value of the *remote* parameter. If both a CLSID and a ProgID are passed to the method, the CLSID takes precedence.

Visual Basic Reference

SetAutoServerSettings Method Example

This example switches a server named "Hello" from local registration to remote, and then back again:

```
Sub SwitchHello()
   Dim oRegClass As New RegClass
   ' Register Hello to run remotely on a machine
 ' called Server1.
   oRegClass.SetAutoServerSettings True, _
  "HelloProj.HelloClass",1 _
   ServerName:="Server1", Protocol:="ncacn_ip_tcp"
   ' Register Hello to run locally again.
   oRegClass.SetAutoServerSettings False, _
  "HelloProj.HelloClass"
End Sub
```

Visual Basic Reference

Visual Studio 6.0

SetData Method (ActiveX Controls)

See Also Example Applies To

Inserts data into a **DataObject** object using the specified data format.

Syntax

object.**SetData** [*data*], [*format*]

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
data	Optional A variant containing the data to be passed to the **DataObject** object.
format	Optional. A constant or value that specifies the format of the data being passed, as described in Settings.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFText	1	Text (.txt files)
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	Metafile (.wmf files)
vbCFEMetafile	14	Enhanced metafile (.emf files)
vbCFDIB	8	Device-independent bitmap (DIB)
vbCFPalette	9	Color palette
vbCFFiles	15	List of files
vbCFRTF	-16639	Rich text format (.rtf files)

Remarks

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

The *data* argument is optional. This allows you to set several different formats that the source component can support without having to load the data separately for each format. Multiple formats are set by calling **SetData** several times, each time using a different format. If you wish to start fresh, use the **Clear** method to clear all data and format information from the **DataObject**.

The *format* argument is also optional, but either the *data* or *format* argument must be specified. If *data* is specified, but not *format*, then Visual Basic will try to determine the format of the data. If it is unsuccessful, then an error is generated. When the target requests the data, and a format was specified, but no data was provided, the sources **OLESetData** event occurs, and the source can then provide the requested data type.

It's possible for the **GetData** and **SetData** methods to use data formats other than those listed in Settings, including user-defined formats registered with Windows via the `RegisterClipboardFormat()` API function. However, there are a few caveats:

- The **SetData** method requires the data to be in the form of a byte array when it does not recognize the data format specified.
- The **GetData** method always returns data in a byte array when it is in a format that it doesn't recognize, although Visual Basic can transparently convert this returned byte array into other data types, such as strings.
- The byte array returned by **GetData** will be larger than the actual data when running on some operating systems, with arbitrary bytes at the end of the array. The reason for this is that Visual Basic does not know the data's format, and knows only the amount of memory that the operating system has allocated for the data. This allocation of memory is often larger than is actually required for the data. Therefore, there may be extraneous bytes near the end of the allocated memory segment. As a result, you must use appropriate functions to interpret the returned data in a meaningful way (such as truncating a string at a particular length with the **Left** function if the data is in a text format).

Visual Basic Reference

Visual Studio 6.0

SetData Method (DataObject Object)

See Also Example Applies To

Inserts data into a **DataObject** object using the specified data format.

Syntax

object.**SetData** [*data*], [*format*]

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
data	Optional A variant containing the data to be passed to the **DataObject** object.
format	Optional. A constant or value that specifies the format of the data being passed, as described in Settings.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFText	1	Text (.txt files)
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	Metafile (.wmf files)
vbCFEMetafile	14	Enhanced metafile (.emf files)
vbCFDIB	8	Device-independent bitmap (DIB)
vbCFPalette	9	Color palette
vbCFFiles	15	List of files
vbCFRTF	-16639	Rich text format (.rtf files)

Remarks

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

The *data* argument is optional. This allows you to set several different formats that the source component can support without having to load the data separately for each format. Multiple formats are set by calling **SetData** several times, each time using a different format. If you wish to start fresh, use the **Clear** method to clear all data and format information from the **DataObject**.

The *format* argument is also optional, but either the *data* or *format* argument must be specified. If *data* is specified, but not *format*, then Visual Basic will try to determine the format of the data. If it is unsuccessful, then an error is generated. When the target requests the data, and a format was specified, but no data was provided, the sources **OLESetData** event occurs, and the source can then provide the requested data type.

It's possible for the **GetData** and **SetData** methods to use data formats other than those listed in Settings, including user-defined formats registered with Windows via the `RegisterClipboardFormat()` API function. However, there are a few caveats:

- The **SetData** method requires the data to be in the form of a byte array when it does not recognize the data format specified.
- The **GetData** method always returns data in a byte array when it is in a format that it doesn't recognize, although Visual Basic can transparently convert this returned byte array into other data types, such as strings.
- The byte array returned by **GetData** will be larger than the actual data when running on some operating systems, with arbitrary bytes at the end of the array. The reason for this is that Visual Basic does not know the data's format, and knows only the amount of memory that the operating system has allocated for the data. This allocation of memory is often larger than is actually required for the data. Therefore, there may be extraneous bytes near the end of the allocated memory segment. As a result, you must use appropriate functions to interpret the returned data in a meaningful way (such as truncating a string at a particular length with the **Left** function if the data is in a text format).

Visual Studio 6.0

Visual Basic: MSChart Control

SetData Method (MSChart)

See Also Example Applies To

Sets the value for a specific data point in the data grid associated with a chart.

Syntax

object.**SetData** *(row, column, dataPoint, nullFlag)*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
row	Integer. Identifies the row containing the data point value.
column	Integer. Identifies the column containing the data point value.
dataPoint	Double. The data point value.
nullFlag	Integer. Indicates whether or not the data point value is a null.

Visual Basic Reference

Visual Studio 6.0

SetData Method

See Also Example Applies To

Puts a picture on the **Clipboard** object using the specified graphic format. Doesn't support named arguments.

Syntax

object.**SetData** *data*, *format*

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
data	Required. A graphic to be placed on the **Clipboard** object.
format	Optional. A constant or value that specifies one of the **Clipboard** object formats recognized by Visual Basic, as described in Settings. If *format* is omitted, **SetData** automatically determines the graphic format.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFBitmap	2	Bitmap (.bmp files)
vbCFMetafile	3	Metafile (.wmf files)
vbCFDIB	8	Device-independent bitmap (DIB)
vbCFPalette	9	Color palette

Remarks

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

You set the graphic that is to be placed onto the **Clipboard** object with either the **LoadPicture** function or the **Picture** property of a **Form**, **Image**, or **PictureBox**.

Visual Basic Extensibility Reference

Visual Studio 6.0

SetFocus Method

See Also Example Applies To Specifics

Moves the focus to the specified window.

Syntax

object.**SetFocus**

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

Remarks

Use the **SetFocus** method on windows that are already visible.

Visual Basic Extensibility Reference

SetFocus Method Example

The following example uses the **SetFocus** method to move the focus to a particular member of the **Windows** collection; that is, it makes that window behave as if you had clicked its title bar with your mouse.

```
Application.VBE.Windows(9).SetFocus
```

Visual Basic Reference

Visual Studio 6.0

SetFocus Method

See Also Example Applies To

Moves the focus to the specified control or form.

Syntax

object.**SetFocus**

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

Remarks

The object must be a **Form** object, **MDIForm** object, or control that can receive the focus. After invoking the **SetFocus** method, any user input is directed to the specified form or control.

You can only move the focus to a visible form or control. Because a form and controls on a form aren't visible until the form's Load event has finished, you can't use the **SetFocus** method to move the focus to the form being loaded in its own Load event unless you first use the **Show** method to show the form before the Form_Load event procedure is finished.

You also can't move the focus to a form or control if the **Enabled** property is set to **False**. If the **Enabled** property has been set to **False** at design time, you must first set it to **True** before it can receive the focus using the **SetFocus** method.

Visual Basic Extensibility Reference

Visual Studio 6.0

SetSelection Method

See Also Example Applies To Specifics

Sets the selection in the code pane.

Syntax

object.**SetSelection(***startline*, *startcol*, *endline*, *endcol***)**

The **SetSelection** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
startline	Required. A Long specifying the first line of the selection.
startcol	Required. A **Long** specifying the first column of the selection.
endline	Required. A **Long** specifying the last line of the selection.
endcol	Required. A **Long** specifying the last column of the selection.

Visual Basic Extensibility Reference

SetSelection Method Example

The following example uses the **SetSelection** method to select the text whose first character is the one immediately after the fourth character on the second line of `CodePanes(1)` and whose last character is the fifteenth character on the third line.

```
Application.VBE.CodePanes(1).SetSelection 2,4,3,15
```

Visual Studio 6.0

Visual Basic: MSChart Control

SetSize Method

See Also Example Applies To

Resizes the number of data columns and rows, as well as the number of levels of column labels and row labels of a data grid associated with a chart at one time.

Syntax

object.**SetSize** (*rowLabelCount*, *columnLabelCount*, *dataRowCount*, *columnLabelCount*)

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
rowLabelCount	Integer. Returns or sets the number of levels of row labels you want on the data grid.
columnLabelCount	Integer. Returns or sets the number of levels of column labels you want on the data grid.
dataRowCount	Integer. Returns or sets the number of data rows you want on the data grid.
columnLabelCount	Integer. Returns or sets the number of data columns you want on the data grid.

Remarks

This method can be used in place of **RowCount**, **ColumnCount**, **RowLabelCount** and **ColumnLabelCount**.

If you reduce the size of the data grid, data in deleted rows or columns is destroyed.

Visual Basic Reference

Visual Studio 6.0

SetText Method

See Also Example Applies To

Puts a text string on the **Clipboard** object using the specified **Clipboard** object format. Doesn't support named arguments.

Syntax

object.**SetText** *data*, *format*

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
data	Required. String data to be placed onto the Clipboard.
Format	Optional. A constant or value that specifies one of the Clipboard formats recognized by Visual Basic, as described in Settings.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFLink	&HBF00	DDE conversation information
vbCFRTF	&HBF01	RichText Format
vbCFText	1	(Default) Text

Remarks

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

Visual Basic Reference

SetText Method Example

This example uses the **SetText** method to copy text from a text box to the Clipboard. To try this example, paste the code into the Declarations section of a form with a text box named Text1, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Const CF_TEXT = 1   ' Define bitmap format.
   Dim I, Msg, Temp   ' Declare variables.
   On Error Resume Next   ' Set up error handling.
   Msg = "Type anything you like into the text box below."
   Text1.Text = InputBox(Msg)   ' Get text from user.
   Msg = "Choose OK to copy the contents of the text box "
   Msg = Msg & "to the Clipboard."
   MsgBox Msg   ' Display message.
   ClipBoard.Clear   ' Clear Clipboard.
   Clipboard.SetText Text1.Text   ' Put text on Clipboard.
   If Clipboard.GetFormat(CF_TEXT) Then
      Text1.Text = ""   ' Clear the text box.
      Msg = "The text is now on the Clipboard. Choose OK "
      Msg = Msg & "to copy the text from the Clipboard back "
      Msg = Msg & "to the text box."
      MsgBox Msg   ' Display message.
      Temp = Clipboard.GetText(CF_TEXT)   ' Get Clipboard text.
      For I = Len(Temp) To 1 Step -1   ' Reverse the text.
         Text1.Text = Text1.Text & Mid(Temp, I, 1)
      Next I
   Else
      Msg = "There is no text on the Clipboard."
      MsgBox Msg   ' Display error message.
   End If
End Sub
```

Visual Basic Reference

Visual Studio 6.0

SetViewport Method

See Also Example Applies To

Sets the left and top coordinates of the **UserDocument** that will be visible in the Viewport.

Syntax

object.**SetViewPort** *left, top*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
left	Required. A value of type Single that specifies the left coordinate of the UserDocument.
top	Required. A value of type Single that specifies the top coordinate of the UserDocument.

Visual Basic: MAPI Controls

Visual Studio 6.0

Show Method (MAPIMessages Control)

See Also Example Applies To

Displays the mail Address Book dialog box or the details of the currently indexed recipient.

Syntax

object.**Show** [*value*]

The **Show** method 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 the type of dialog box to show, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	Displays a dialog box that shows the details of the currently indexed recipient. The amount of information presented in the dialog box is determined by the message system. At a minimum, it contains the display name and address of the recipient.
False	(Default) Displays the mail **Address Book** dialog box. You can use the address book to create or modify a recipient set. Any changes to the address book outside of the compose buffer are not saved.

Visual Basic Extensibility Reference

Visual Studio 6.0

Show Method (VBA Add-In Object Model)

See Also Example Applies To Specifics

Makes the specified code pane the visible code pane in its window.

Syntax

object.**Show**

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

Remarks

The **Show** method makes the specified code pane the pane with the focus in its window.

Visual Basic Extensibility Reference

Show Method Example

The following example uses the **Show** method to move the specified code pane to the foreground.

```
Application.VBE.CodePanes(2).Show
```

Visual Basic Reference

Visual Studio 6.0

Show Method

See Also Example Applies To

Displays an **MDIForm** or **Form** object. Doesn't support named arguments.

Syntax

object.**Show** *style, ownerform*

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

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form associated with the active form module is assumed to be *object*.
style	Optional. Integer that determines if the form is modal or modeless. If *style* is 0, the form is modeless; if *style* is 1, the form is modal.
ownerform	Optional. A string expression that specifies the component which "owns" the form being shown. For standard Visual Basic forms, use the keyword **Me**

Remarks

If the specified form isn't loaded when the **Show** method is invoked, Visual Basic automatically loads it.

When **Show** displays a modeless form, subsequent code is executed as it's encountered. When **Show** displays a modal form, no subsequent code is executed until the form is hidden or unloaded.

When **Show** displays a modal form, no input (keyboard or mouse click) can occur except to objects on the modal form. The program must hide or unload a modal form (usually in response to some user action) before input to another form can occur. An **MDIForm** can't be modal.

Although other forms in your application are disabled when a modal form is displayed, other applications aren't.

The startup form of an application is automatically shown after its Load event is invoked.

Here is an example of how the *ownerform* argument is used with the **Show** method:

```
Private Sub cmdShowResults_Click()
   ' Show a modal form named frmResults.
   frmResults.Show vbModal, Me
End Sub
```

Visual Basic Reference

Show Method Example

This example uses the **Show** method to show a hidden form. To try this example, paste the code into the Declarations section of a non-MDI form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Msg   ' Declare variable.
   Hide   ' Hide form.
   Msg = "Choose OK to make the form reappear."
   MsgBox Msg   ' Display message.
   Show   ' Show form again.
End Sub
```

Visual Basic: CommonDialog Control

Visual Studio 6.0

ShowColor Method

See Also Example Applies To

Displays the **CommonDialog** control's Color dialog box.

Syntax

object.**ShowColor**

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

Visual Basic: CommonDialog Control

ShowColor, ShowFont, ShowHelp, ShowOpen, ShowPrinter, ShowSave Methods Example

This example uses the **CommonDialog** control and the **ShowColor**, **ShowFont**, **ShowHelp**, **ShowOpen**, **ShowPrinter**, and **ShowSave** methods to display the common dialog boxes. To try this example, paste the code into the Declarations section of a form with **CommandButton**, **OptionButton** (set the option button's **Index** property to 0), and **CommonDialog** controls. Press F5 and select the option button for the common dialog box you want and choose the command button.

```
Private Sub Form_Paint ()
   Static FlagFormPainted As Integer
   ' When form is painting for first time,
   If FlagFormPainted <> True Then
      For i = 1 To 5
         Load Option1(i)   ' Add five option buttons to array.
         Option1(i).Top = Option1(i - 1).Top + 350
         Option1(i).Visible = True
      Next i
      Option1(0).Caption = "Open"   ' Put caption on each option button.
      Option1(1).Caption = "Save"
      Option1(2).Caption = "Color"
      Option1(3).Caption = "Font"
       Option1(4).Caption = "Printer"
      Option1(5).Caption = "Help"
      Command1.Caption = "Show Dlg"   ' Label command button.
      FlagFormPainted = True   ' Form is done painting.
   End If
End Sub

Private Sub Command1_Click ()
   If Option1(0).Value Then   ' If Open option button selected,
      CommonDialog1.ShowOpen   ' display Open common dialog box.
   ElseIf Option1(1).Value Then   ' Or,
      CommonDialog1.ShowSave   ' display Save common dialog box.
   ElseIf Option1(2).Value Then   ' Or,
      CommonDialog1.ShowColor   ' display Color common dialog box.
   ElseIf Option1(3).Value Then   ' Or,
      CommonDialog1.Flags = cdlCFBoth   ' Flags property must be set
            ' to cdlCFBoth,                    ' cdlCFPrinterFonts,
            ' or cdlCFScreenFonts before               ' using ShowFont method.
      CommonDialog1.ShowFont   ' Display Font common dialog box.
   ElseIf Option1(4).Value Then   ' Or,
      CommonDialog1.ShowPrinter   ' display Printer common dialog box.
   ElseIf Option1(5).Value Then   ' Or,
      CommonDialog1.HelpFile = "VB5.hlp"
      CommonDialog1.HelpCommand = cdlHelpContents
      CommonDialog1.ShowHelp   ' Display Visual Basic Help contents topic.
      End If
End Sub
```

Visual Basic: CommonDialog Control

Visual Studio 6.0

ShowFont Method

See Also Example Applies To

Displays the **CommonDialog** control's Font dialog box.

Syntax

object.**ShowFont**

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

Remarks

Before you use the **ShowFont** method, you must set the **Flags** property of the **CommonDialog** control to one of three constants or values: **cdlCFBoth** or &H3, **cdlCFPrinterFonts** or &H2, or **cdlCFScreenFonts** or &H1. If you don't set **Flags**, a message box is displayed advising you that "There are no fonts installed," and a run-time error occurs.

Visual Basic: CommonDialog Control

ShowColor, ShowFont, ShowHelp, ShowOpen, ShowPrinter, ShowSave Methods Example

This example uses the **CommonDialog** control and the **ShowColor**, **ShowFont**, **ShowHelp**, **ShowOpen**, **ShowPrinter**, and **ShowSave** methods to display the common dialog boxes. To try this example, paste the code into the Declarations section of a form with **CommandButton**, **OptionButton** (set the option button's **Index** property to 0), and **CommonDialog** controls. Press F5 and select the option button for the common dialog box you want and choose the command button.

```
Private Sub Form_Paint ()
   Static FlagFormPainted As Integer
   ' When form is painting for first time,
   If FlagFormPainted <> True Then
      For i = 1 To 5
         Load Option1(i)   ' Add five option buttons to array.
         Option1(i).Top = Option1(i - 1).Top + 350
         Option1(i).Visible = True
      Next i
      Option1(0).Caption = "Open"   ' Put caption on each option button.
      Option1(1).Caption = "Save"
      Option1(2).Caption = "Color"
      Option1(3).Caption = "Font"
       Option1(4).Caption = "Printer"
      Option1(5).Caption = "Help"
      Command1.Caption = "Show Dlg"   ' Label command button.
      FlagFormPainted = True   ' Form is done painting.
   End If
End Sub

Private Sub Command1_Click ()
   If Option1(0).Value Then   ' If Open option button selected,
      CommonDialog1.ShowOpen   ' display Open common dialog box.
   ElseIf Option1(1).Value Then   ' Or,
      CommonDialog1.ShowSave   ' display Save common dialog box.
   ElseIf Option1(2).Value Then   ' Or,
      CommonDialog1.ShowColor   ' display Color common dialog box.
   ElseIf Option1(3).Value Then   ' Or,
      CommonDialog1.Flags = cdlCFBoth   ' Flags property must be set
            ' to cdlCFBoth,                    ' cdlCFPrinterFonts,
            ' or cdlCFScreenFonts before               ' using ShowFont method.
      CommonDialog1.ShowFont   ' Display Font common dialog box.
   ElseIf Option1(4).Value Then   ' Or,
      CommonDialog1.ShowPrinter   ' display Printer common dialog box.
   ElseIf Option1(5).Value Then   ' Or,
      CommonDialog1.HelpFile = "VB5.hlp"
      CommonDialog1.HelpCommand = cdlHelpContents
      CommonDialog1.ShowHelp   ' Display Visual Basic Help contents topic.
      End If
End Sub
```

Visual Basic: CommonDialog Control

Visual Studio 6.0

ShowHelp Method

See Also Example Applies To

Runs Winhlp32.exe and displays the Help file you specify.

Syntax

object.**ShowHelp**

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

Remarks

Before you use the **ShowHelp** method, you must set the **HelpFile** and **HelpCommand** properties of the **CommonDialog** control to one of their appropriate constants or values. Otherwise, Winhlp32.exe doesn't display the Help file.

Visual Basic: CommonDialog Control

ShowColor, ShowFont, ShowHelp, ShowOpen, ShowPrinter, ShowSave Methods Example

This example uses the **CommonDialog** control and the **ShowColor**, **ShowFont**, **ShowHelp**, **ShowOpen**, **ShowPrinter**, and **ShowSave** methods to display the common dialog boxes. To try this example, paste the code into the Declarations section of a form with **CommandButton**, **OptionButton** (set the option button's **Index** property to 0), and **CommonDialog** controls. Press F5 and select the option button for the common dialog box you want and choose the command button.

```
Private Sub Form_Paint ()
   Static FlagFormPainted As Integer
   ' When form is painting for first time,
   If FlagFormPainted <> True Then
      For i = 1 To 5
         Load Option1(i)   ' Add five option buttons to array.
         Option1(i).Top = Option1(i - 1).Top + 350
         Option1(i).Visible = True
      Next i
      Option1(0).Caption = "Open"   ' Put caption on each option button.
      Option1(1).Caption = "Save"
      Option1(2).Caption = "Color"
      Option1(3).Caption = "Font"
       Option1(4).Caption = "Printer"
      Option1(5).Caption = "Help"
      Command1.Caption = "Show Dlg"   ' Label command button.
      FlagFormPainted = True   ' Form is done painting.
   End If
End Sub

Private Sub Command1_Click ()
   If Option1(0).Value Then   ' If Open option button selected,
      CommonDialog1.ShowOpen   ' display Open common dialog box.
   ElseIf Option1(1).Value Then   ' Or,
      CommonDialog1.ShowSave   ' display Save common dialog box.
   ElseIf Option1(2).Value Then   ' Or,
      CommonDialog1.ShowColor   ' display Color common dialog box.
   ElseIf Option1(3).Value Then   ' Or,
      CommonDialog1.Flags = cdlCFBoth   ' Flags property must be set
            ' to cdlCFBoth,                    ' cdlCFPrinterFonts,
            ' or cdlCFScreenFonts before               ' using ShowFont method.
      CommonDialog1.ShowFont   ' Display Font common dialog box.
   ElseIf Option1(4).Value Then   ' Or,
      CommonDialog1.ShowPrinter   ' display Printer common dialog box.
   ElseIf Option1(5).Value Then   ' Or,
      CommonDialog1.HelpFile = "VB5.hlp"
      CommonDialog1.HelpCommand = cdlHelpContents
      CommonDialog1.ShowHelp   ' Display Visual Basic Help contents topic.
      End If
End Sub
```

Visual Basic: CommonDialog Control

Visual Studio 6.0

ShowOpen Method

See Also Example Applies To

Displays the **CommonDialog** control's Open dialog box.

Syntax

object.**ShowOpen**

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

Visual Basic: CommonDialog Control

ShowColor, ShowFont, ShowHelp, ShowOpen, ShowPrinter, ShowSave Methods Example

This example uses the **CommonDialog** control and the **ShowColor**, **ShowFont**, **ShowHelp**, **ShowOpen**, **ShowPrinter**, and **ShowSave** methods to display the common dialog boxes. To try this example, paste the code into the Declarations section of a form with **CommandButton**, **OptionButton** (set the option button's **Index** property to 0), and **CommonDialog** controls. Press F5 and select the option button for the common dialog box you want and choose the command button.

```
Private Sub Form_Paint ()
   Static FlagFormPainted As Integer
   ' When form is painting for first time,
   If FlagFormPainted <> True Then
      For i = 1 To 5
         Load Option1(i)   ' Add five option buttons to array.
         Option1(i).Top = Option1(i - 1).Top + 350
         Option1(i).Visible = True
      Next i
      Option1(0).Caption = "Open"   ' Put caption on each option button.
      Option1(1).Caption = "Save"
      Option1(2).Caption = "Color"
      Option1(3).Caption = "Font"
       Option1(4).Caption = "Printer"
      Option1(5).Caption = "Help"
      Command1.Caption = "Show Dlg"   ' Label command button.
      FlagFormPainted = True   ' Form is done painting.
   End If
End Sub

Private Sub Command1_Click ()
   If Option1(0).Value Then   ' If Open option button selected,
      CommonDialog1.ShowOpen   ' display Open common dialog box.
   ElseIf Option1(1).Value Then   ' Or,
      CommonDialog1.ShowSave   ' display Save common dialog box.
   ElseIf Option1(2).Value Then   ' Or,
      CommonDialog1.ShowColor   ' display Color common dialog box.
   ElseIf Option1(3).Value Then   ' Or,
      CommonDialog1.Flags = cdlCFBoth   ' Flags property must be set
            ' to cdlCFBoth,                    ' cdlCFPrinterFonts,
            ' or cdlCFScreenFonts before               ' using ShowFont method.
      CommonDialog1.ShowFont   ' Display Font common dialog box.
   ElseIf Option1(4).Value Then   ' Or,
      CommonDialog1.ShowPrinter   ' display Printer common dialog box.
   ElseIf Option1(5).Value Then   ' Or,
      CommonDialog1.HelpFile = "VB5.hlp"
      CommonDialog1.HelpCommand = cdlHelpContents
      CommonDialog1.ShowHelp   ' Display Visual Basic Help contents topic.
      End If
End Sub
```

Visual Basic: CommonDialog Control

Visual Studio 6.0

ShowPrinter Method

See Also Example Applies To

Displays the **CommonDialog** control's Printer dialog box.

Syntax

object.**ShowPrinter**

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

Visual Basic: CommonDialog Control

ShowColor, ShowFont, ShowHelp, ShowOpen, ShowPrinter, ShowSave Methods Example

This example uses the **CommonDialog** control and the **ShowColor**, **ShowFont**, **ShowHelp**, **ShowOpen**, **ShowPrinter**, and **ShowSave** methods to display the common dialog boxes. To try this example, paste the code into the Declarations section of a form with **CommandButton**, **OptionButton** (set the option button's **Index** property to 0), and **CommonDialog** controls. Press F5 and select the option button for the common dialog box you want and choose the command button.

```
Private Sub Form_Paint ()
   Static FlagFormPainted As Integer
   ' When form is painting for first time,
   If FlagFormPainted <> True Then
      For i = 1 To 5
         Load Option1(i)   ' Add five option buttons to array.
         Option1(i).Top = Option1(i - 1).Top + 350
         Option1(i).Visible = True
      Next i
      Option1(0).Caption = "Open"   ' Put caption on each option button.
      Option1(1).Caption = "Save"
      Option1(2).Caption = "Color"
      Option1(3).Caption = "Font"
       Option1(4).Caption = "Printer"
      Option1(5).Caption = "Help"
      Command1.Caption = "Show Dlg"   ' Label command button.
      FlagFormPainted = True   ' Form is done painting.
   End If
End Sub

Private Sub Command1_Click ()
   If Option1(0).Value Then   ' If Open option button selected,
      CommonDialog1.ShowOpen   ' display Open common dialog box.
   ElseIf Option1(1).Value Then   ' Or,
      CommonDialog1.ShowSave   ' display Save common dialog box.
   ElseIf Option1(2).Value Then   ' Or,
      CommonDialog1.ShowColor   ' display Color common dialog box.
   ElseIf Option1(3).Value Then   ' Or,
      CommonDialog1.Flags = cdlCFBoth   ' Flags property must be set
            ' to cdlCFBoth,                      ' cdlCFPrinterFonts,
            ' or cdlCFScreenFonts before                ' using ShowFont method.
      CommonDialog1.ShowFont   ' Display Font common dialog box.
   ElseIf Option1(4).Value Then   ' Or,
      CommonDialog1.ShowPrinter   ' display Printer common dialog box.
   ElseIf Option1(5).Value Then   ' Or,
      CommonDialog1.HelpFile = "VB5.hlp"
      CommonDialog1.HelpCommand = cdlHelpContents
      CommonDialog1.ShowHelp   ' Display Visual Basic Help contents topic.
      End If
End Sub
```

Visual Basic: CommonDialog Control

Visual Studio 6.0

ShowSave Method

See Also Example Applies To

Displays the **CommonDialog** control's Save As dialog box.

Syntax

object.**ShowSave**

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

Visual Basic: CommonDialog Control

ShowColor, ShowFont, ShowHelp, ShowOpen, ShowPrinter, ShowSave Methods Example

This example uses the **CommonDialog** control and the **ShowColor**, **ShowFont**, **ShowHelp**, **ShowOpen**, **ShowPrinter**, and **ShowSave** methods to display the common dialog boxes. To try this example, paste the code into the Declarations section of a form with **CommandButton**, **OptionButton** (set the option button's **Index** property to 0), and **CommonDialog** controls. Press F5 and select the option button for the common dialog box you want and choose the command button.

```
Private Sub Form_Paint ()
   Static FlagFormPainted As Integer
   ' When form is painting for first time,
   If FlagFormPainted <> True Then
      For i = 1 To 5
         Load Option1(i)   ' Add five option buttons to array.
         Option1(i).Top = Option1(i - 1).Top + 350
         Option1(i).Visible = True
      Next i
      Option1(0).Caption = "Open"   ' Put caption on each option button.
      Option1(1).Caption = "Save"
      Option1(2).Caption = "Color"
      Option1(3).Caption = "Font"
       Option1(4).Caption = "Printer"
      Option1(5).Caption = "Help"
      Command1.Caption = "Show Dlg"   ' Label command button.
      FlagFormPainted = True   ' Form is done painting.
   End If
End Sub

Private Sub Command1_Click ()
   If Option1(0).Value Then   ' If Open option button selected,
      CommonDialog1.ShowOpen   ' display Open common dialog box.
   ElseIf Option1(1).Value Then   ' Or,
      CommonDialog1.ShowSave   ' display Save common dialog box.
   ElseIf Option1(2).Value Then   ' Or,
      CommonDialog1.ShowColor   ' display Color common dialog box.
   ElseIf Option1(3).Value Then   ' Or,
      CommonDialog1.Flags = cdlCFBoth   ' Flags property must be set
            ' to cdlCFBoth,                    ' cdlCFPrinterFonts,
            ' or cdlCFScreenFonts before               ' using ShowFont method.
      CommonDialog1.ShowFont   ' Display Font common dialog box.
   ElseIf Option1(4).Value Then   ' Or,
      CommonDialog1.ShowPrinter   ' display Printer common dialog box.
   ElseIf Option1(5).Value Then   ' Or,
      CommonDialog1.HelpFile = "VB5.hlp"
      CommonDialog1.HelpCommand = cdlHelpContents
      CommonDialog1.ShowHelp   ' Display Visual Basic Help contents topic.
      End If
End Sub
```

Visual Basic Reference

Visual Studio 6.0

ShowWhatsThis Method

See Also Example Applies To

Displays a selected topic in a Help file using the What's This popup provided by Help in 32-bit Windows operating systems.

Syntax

object.**ShowWhatsThis**

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

Remarks

The **ShowWhatsThis** method is very useful for providing context-sensitive Help from a context menu in your application. The method displays the topic identified by the **WhatsThisHelpID** property of the object specified in the syntax.

Visual Basic Reference

ShowWhatsThis Method Example

This example displays the What's This Help topic for a **CommandButton** control by selecting a menu command from a context menu created for the button. Set the **WhatsThisHelp** property of the form to **True**. Place a **CommandButton** control on a form, create a menu using the Menu Editor with a top-level invisible item named mnuBtnContextMenu, and a sub-menu named mnuBtnWhatsThis with a caption of "What's This?".

```
Private ThisControl As Control

Private Sub Command1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
      If Button = vbRightButton Then
         Set ThisControl = Command1
         PopupMenu mnuBtnContextMenu
      End If
   Set ThisControl = Nothing
End Sub

Private Sub mnuBtnWhatsThis_Click()
   ThisControl.ShowWhatsThis
End Sub
```

Visual Basic: MAPI Controls

Visual Studio 6.0

SignOff Method

See Also Example Applies To

Ends the messaging session and signs the user off from the account specified by the **UserName** and **Password** properties.

Syntax

object.**SignOff**

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

Visual Basic: MAPI Controls

Visual Studio 6.0

SignOn Method

See Also Example Applies To

Logs the user into the account specified by the **UserName** and **Password** properties, and provides a session handle to the underlying message subsystem.

Syntax

object.**SignOn**

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

Remarks

The session handle is stored in the **SessionID** property. Depending on the value of the **NewSession** property, the session handle may refer to a newly created session or an existing session.

Visual Basic Reference

Visual Studio 6.0

Size Method

See Also Example Applies To

Changes the width and height of a **UserControl** object.

Syntax

object.**Size** *width, height*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
width	Required. The width in twips of the object.
height	Required. The height in twips of the object.

Remarks

The **Width** and **Height** properties of a **UserControl** object are always given in Twips, regardless of **ScaleMode**.

Visual Basic for Applications Reference

Visual Studio 6.0

Skip Method

See Also Example Applies To Specifics

Description

Skips a specified number of characters when reading a **TextStream** file.

Syntax

object.**Skip(***characters***)**

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

Part	Description
object	Required. Always the name of a **TextStream** object.
characters	Required. Number of characters to skip when reading a file.

Remarks

Skipped characters are discarded.

Visual Basic for Applications Reference

Visual Studio 6.0

SkipLine Method

See Also Example Applies To Specifics

Description

Skips the next line when reading a **TextStream** file.

Syntax

object.**SkipLine**

The *object* is always the name of a **TextStream** object.

Remarks

Skipping a line means reading and discarding all characters in a line up to and including the next newline character.

An error occurs if the file is not open for reading.

Visual Basic: RichTextBox Control

Visual Studio 6.0

Span Method

See Also Example Applies To

Selects text in a **RichTextBox** control based on a set of specified characters.

Syntax

object.**Span** *characterset*, *forward*, *negate*

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
characterset	Required. A string expression that specifies the set of characters to look for when extending the selection, based on the value of *negate*.
forward	Optional. A Boolean expression that determines which direction the insertion point moves, as described in Settings.
negate	Optional. A Boolean expression that determines whether the characters in *characterset* define the set of target characters or are excluded from the set of target characters, as described in Settings.

Settings

The settings for *forward* are:

Setting	Description
True	(Default) Selects text from the current insertion point or the beginning of the current selection forward, toward the end of the text.
False	Selects text from the current insertion point or the beginning of the current selection backward, toward the start of the text.

The settings for *negate* are:

Setting	Description
True	The characters included in the selection are those which do not appear in the *characterset* argument. The selection stops at the first character found that appears in the *characterset* argument.
False	(Default) The characters included in the selection are those which appear in the *characterset* argument. The selection stops at the first character found that does not appear in the *characterset* argument.

Remarks

The **Span** method is primarily used to easily select a word or sentence in the **RichTextBox** control.

If the **Span** method cannot find the specified characters based on the values of the arguments, then the current insertion point or selection remains unchanged.

The **Span** method does not return any data.

Visual Basic: RichTextBox Control

Span Method Example

This example defines a pair of keyboard shortcuts that selects text in a **RichTextBox** control to the end of a sentence (CTRL+S) or the end of a word (CTRL+W). To try this example, put a **RichTextBox** control on a form. Paste this code into the KeyUp event of the **RichTextBox** control. Then run the example.

```
Private Sub RichTextBox1_KeyUp (KeyCode As Integer, Shift As Integer)
   If Shift = vbCtrlMask Then
      Select Case KeyCode
         ' If Ctrl+S:
         Case vbKeyS
            ' Select to the end of the sentence.
            RichTextBox1.Span ".?!:", True, True
            ' Extend selection to include punctuation.
            RichTextBox1.SelLength = RichTextBox1.SelLength + 1
         ' If Ctrl+W:
         Case vbKeyW
            ' Select to the end of the word.
            RichTextBox1.Span " ,;:.?!", True, True
      End Select
   End If
End Sub
```

Visual Basic: DataGrid Control

Visual Studio 6.0

SplitContaining Method

See Also Example Applies To

Returns the **Index** value of the split containing the specified coordinate pair.

Syntax

object.**SplitContaining** *x, y*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
x	Required. A single precision value that defines the horizontal coordinate, based on the coordinate system of the grid's container.
y	Required. A single precision value that defines the vertical coordinate, based on the coordinate system of the grid's container.

Remarks

This value ranges from 0 to 1 less than the setting of the **Count** property of the Splits collection (0 to **Splits.Count** - 1).

This method is useful when working with mouse and drag events when you are trying to determine where the user clicked or dropped another control in terms of a grid column.

If either argument is outside of the grid's data area, this method returns -1.

Visual Basic: Windows Controls

Visual Studio 6.0

StartLabelEdit Method

See Also Example Applies To

Enables a user to edit a label.

Syntax

object.**StartLabelEdit**

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

Remarks

The **StartLabelEdit** method must be used to initiate a label editing operation when the **LabelEdit** property is set to 1 (Manual).

When the **StartLabelEdit** method is invoked upon an object, the BeforeLabelEdit event is also generated.

Visual Basic: Windows Controls

StartLabelEdit Method Example

This example adds several **Node** objects to a **TreeView** control. After a **Node** is selected, click the form to begin editing it. To try the example, place a **TreeView** control on a form, and paste the code into the form's Declarations section. Run the example, select a **Node**, and click the form.

```
Private Sub Form_Load
   Dim nodX As Node

   Set nodX = TreeView1.Nodes.Add(,,,"Da Vinci") ' Root
   Set nodX = TreeView1.Nodes.Add(1,tvwChild,,"Titian")
   Set nodX = TreeView1.Nodes.Add(1,tvwChild,,"Rembrandt")
   Set nodX = TreeView1.Nodes.Add(1,tvwChild,,"Goya")
   Set nodX = TreeView1.Nodes.Add(1,tvwChild,,"David")
   nodX.EnsureVisible ' Expand tree to see all nodes.
End Sub

Private Sub Form_Click()
   ' If selected Node isn't the Root node then allow edits.
   If TreeView1.SelectedItem.Index <> 1 Then
      TreeView1.StartLabelEdit
   End If
End Sub
```

Visual Basic Reference

Visual Studio 6.0

StartLogging Method

See Also Example Applies To

Sets the log target and log mode of an operation.

Syntax

object.**StartLogging** *logTarget, logMode*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
logTarget	Path and filename of the file used to capture output from the **LogEvent** method.
logMode	A value which determines how logging (through the **LogEvent** method) will be carried out. See Settings below.

Settings

The settings for *logMode* are:

Constant	Value	Description
vbLogAuto	0	If running on Windows 95 or later, this option logs messages to the file specified in the **LogFile** property. If running on Windows NT, messages are logged to the Windows NT Application Event Log, with "VBRunTime" used as the application source and App.Title appearing in the description.
VbLogOff	1	Turns all logging off. Messages from UI shunts as well as from the **LogEvent** method are ignored and discarded.
VbLogToFile	2	Forces logging to a file. If no valid filename is present in **LogPath**, logging is ignored, and the property is set to **vbLogOff**.
VbLogToNT	3	Forces logging to the NT event log. If not running on Windows NT, or the event log is unavailable, logging is ignored and the property is set to **vbLogOff**.
VbLogOverwrite	16	Indicates that the logfile should be recreated each time the application starts. This value can be combined with other mode options using the **OR** operator. The default action for

		logging is to append to the existing file. In the case of NT event logging, this flag has no meaning.
VbLogThreadID	32	Indicates that the current thread ID be prepended to the message, in the form "[T:0nnn] ". This value can be combined with other mode options using the **OR** operator. The default action is to show the thread ID only when the application is multi-threaded (either explicitly marked as thread-safe, or implemented as an implicit multithreaded app, such as a local server with the instancing property set to Single-Use, multithreaded).

Visual Basic: Windows Controls

Visual Studio 6.0

Stop Method (Animation Control)

See Also Example Applies To

Stops the play of an .avi file in the **Animation** control.

Syntax

object.**Stop**

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

Remarks

The **Stop** method stops only an animation that was started with the **Play** method. Attempting to use the **Stop** method when the **Autoplay** property is set to True returns an error (35759).