Visual Basic for Applications Reference

Visual Studio 6.0

GetAbsolutePathName Method

See Also Example Applies To Specifics

Description

Returns a complete and unambiguous path from a provided path specification.

Syntax

object.**GetAbsolutePathName(***pathspec***)**

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

Part	Description
object	Required. Always the name of a **FileSystemObject**.
pathspec	Required. Path specification to change to a complete and unambiguous path.

Remarks

A path is complete and unambiguous if it provides a complete reference from the root of the specified drive. A complete path can only end with a path separator character (****) if it specifies the root folder of a mapped drive.

Assuming the current directory is c:\mydocuments\reports, the following table illustrates the behavior of the **GetAbsolutePathName** method.

pathspec	**Returned path**
"c:"	"c:\mydocuments\reports"
"c:.."	"c:\mydocuments"
"c:\\\"	"c:\"
"c:*.*\may97"	"c:\mydocuments\reports*.*\may97"
"region1"	"c:\mydocuments\reports\region1"
"c:\..\..\mydocuments"	"c:\mydocuments"

Visual Basic for Applications Reference

Visual Studio 6.0

GetBaseName Method

See Also Example Applies To Specifics

Description

Returns a string containing the base name of the last component, less any file extension, in a path.

Syntax

object.**GetBaseName(***path***)**

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

Part	Description
object	Required. Always the name of a **FileSystemObject**.
path	Required. The path specification for the component whose base name is to be returned.

Remarks

The **GetBaseName** method returns a zero-length string ("") if no component matches the *path* argument.

Note The **GetBaseName** method works only on the provided *path* string. It does not attempt to resolve the path, nor does it check for the existence of the specified path.

Visual Basic: DataGrid Control

Visual Studio 6.0

GetBookmark Method

See Also Example Applies To

Returns a value containing a bookmark for a row relative to the current row in a **DataGrid** control. Doesn't support named arguments.

Syntax

object.**GetBookmark** *value*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	Required. A long numeric expression that addresses rows of the **DataGrid** control relative to the current row, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
0	Returns bookmark of the current row the same as `DataGrid1.Bookmark`.
1	Returns bookmark of the row following current row.
-1	Returns bookmark of the row preceding current row.
n	Returns bookmark of the row relative to current row based on (`DataGrid1.Row + n`)

Remarks

The **GetBookmark** method may return values that are very different from the **RowBookmark** method because the current row may not be visible.

Visual Basic: DataGrid Control

GetBookmark Method Example

This example checks the updated value of a particular column to make sure that the new value lies between the values of the previous and the next rows.

```
Sub DataGrid1_BeforeColUpdate (ColIndex As Integer, _
 OldValue as Variant, PrevVal, NextVal, CurVal, Cancel As Integer)
   If ColIndex = 1 Then
      PrevVal = DataGrid1.Columns(1).CellValue(_
       DataGrid1.GetBookmark(-1))
      NextVal = DataGrid1.Columns(1).CellValue(_
       DataGrid1.GetBookmark(1))
      CurVal = DataGrid1.Columns(1).Value
      If CurVal > PrevVal Or CurVal < NextVal  Then
         Cancel = True
         MsgBox "Value must be between" & PrevVal _
          & " and " & NextVal
      End If
   End If
End Sub
```

Visual Basic: RDO Data Control

Visual Studio 6.0

GetClipString Method

See Also Example Applies To

The **GetClipString** method returns a delimited string for '*n*' rows in a result set.

Syntax

ResulsetString = *object*.**GetClipString** (*NumRows*, [*ColumnDelimiter*],[*RowDelimiter*], *NullExpr*])

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

Part	Description
ResultsetString	A variable used to reference the entire result set as a delimited string.
Object	An object expression that evaluates to an **rdoResultset** object.
NumRows	Required: **Long** value. Number of rows to copy into the clip string.
ColumnDelimiter	Optional: **Variant**(**String)** expression used to separate data columns as described in Settings. Default is Tab (**VbTab**).
RowDelimiter	Optional: **Variant**(**String)**expression used to separate data rows as described in Settings. Default is carriage return (**VbCr**).
NullExpr	Optional: **Variant**(**String)**expression used when NULL values are encountered as described in Settings. Default is an empty string.

Settings

The row and column delimiters can be any length, but are generally one or two bytes long. Generally, the ***ResultsetString*** delimiters are determined by the **Clip** property of the target object. For example, if the string is applied to a grid control, columns are separated by tabs and the rows are separated by carriage returns (the default settings).

The ***NullExpr*** is used to substitute a suitable value in place of NULL values returned from the query. Generally, an empty string or <null> is used.

Remarks

The **GetClipString** method returns a delimited string for '*n*' rows in a result set based on the ***NumRows*** argument. If more rows are requested than are available, only the available rows are returned. Use the **RowCount** property to determine how many rows are actually fetched. The number of rows that can be fetched is constrained by available memory and should be

chosen to suit your application. Don't expect to use **GetClipString** to bring your entire table or result set into memory if it is a large table.

Generally, **GetClipString** works just like the **GetRows** method except that the data is returned as a string instead of a 2-dimensional variant array. **GetClipString** can be used fill a grid control, or any control that has a **Clip** property. It can also be used to format export data from a result set to a sequential file.

After a call to **GetClipString**, the current row is positioned at the next unread row. That is, **GetClipString** is equivalent to using the **Move** (rows) method.

If you are trying to fetch all the rows using multiple **GetClipString** calls, use the **EOF** property to determine if there are rows available. **GetClipString** returns less than the number requested either at the end of the **rdoResultset**, or if it cannot fetch a row in the range requested. For example, if a fifth row cannot be retrieved in a group of ten rows that you're trying to fetch, **GetClipString** returns four rows and leaves currency on the row that caused the problem. It will not generate a run-time error.

The ***ColumnDelimiter*** optional parameter can be used to substitute a different column delimiter than the default tab (**Chr**$(9)) character, and the ***RowDelimiter*** optional parameter can be used to substitute a different row delimiter. This is useful when working with a control that accepts a clip format, but requires different characters for the column and row delimiters (some grids have been known to require both a carriage return and a line feed character for a row delimiter).

Visual Basic: RDO Data Control

GetClipString Example (Remote Data)

The following example creates a clip string from a result set containing a selected set of rows from the Publishers table and fills a **Grid** control from the string by applying it to the **Clip** property.

```
Dim rs As rdoResultset
Set rs = MyConnection.OpenResultset( _
"Select * from Publishers Where State = WA", _
   rdOpenNone)
MyGrid.Rows = rs.RowCount
MyGrid.Cols = rs.rdoColumns.Count
MyGrid.SelStartRow = 1
MyGrid.SelEndRow = MyGrid.Rows
MyGrid.SelStartCol = 0
MyGrid.SelEndCol = MyGrid.Cols - 1
MyGrid.Clip = rs.GetClipString(rs.RowCount)
```

Visual Basic Reference

Visual Studio 6.0

GetData Method (ActiveX Controls)

See Also Example Applies To

Returns a graphic from the **Clipboard** object. Doesn't support named arguments.

Syntax

object.**GetData (***format***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	Optional. A constant or value that specifies the **Clipboard** graphics format, as described in Settings. Parentheses must enclose the constant or value. If *format* is 0 or omitted, **GetData** automatically uses the appropriate 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.

If no graphic on the **Clipboard** object matches the expected format, nothing is returned. If only a color palette is present on the **Clipboard** object, a minimum size (1 x 1) DIB is created.

Visual Basic Reference

Visual Studio 6.0

GetData Method (DataObject Object)

See Also Example Applies To

Returns data from a **DataObject** object in the form of a variant.

Syntax

object.**GetData (***format***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	A constant or value that specifies the data format, as described in Settings. Parentheses must enclose the constant or value. If *format* is 0 or omitted, **GetData** automatically uses the appropriate format.

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.

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).

Note Not all applications support **vbcfBitmap** or **vbCFPalette**, so it is recommended that you use **vbCFDIB** whenever possible.

Visual Studio 6.0

Visual Basic: MSChart Control

GetData Method (MSChart)

See Also Example Applies To

Returns the value currently stored in a specific data point in the data grid associated with a chart.

Syntax

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

The **GetData** 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: Winsock Control

Visual Studio 6.0

GetData Method (WinSock Control)

See Also Example Applies To

Retrieves the current block of data and stores it in a variable of type variant.

Return Value

Void

Syntax

object.**GetData** *data*, [*type*,] [*maxLen*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
data	Where retrieved data will be stored after the method returns successfully. If there is not enough data available for requested type, *data* will be set to Empty.
type	Optional. Type of data to be retrieved, as shown in Settings.
maxLen	Optional. Specifies the desired size when receiving a byte array or a string. If this parameter is missing for byte array or string, all available data will be retrieved. If provided for data types other than byte array and string, this parameter is ignored.

Settings

The settings for *type* are:

Description	Constant
Byte	**vbByte**
Integer	**vbInteger**
Long	**vbLong**
Single	**vbSingle**

Double	**vbDouble**
Currency	**vbCurrency**
Date	**vbDate**
Boolean	**vbBoolean**
SCODE	**vbError**
String	**vbString**
Byte Array	**vbArray + vbByte**

Remarks

It's common to use the **GetData** method with the DataArrival event, which includes the *totalBytes* argument. If you specify a *maxlen* that is less than the *totalBytes* argument, you will get the warning 10040 indicating that the remaining bytes will be lost.

Visual Basic: Winsock Control

GetData Method (WinSock Control), DataArrival Event Example

The example uses the **GetData** method in the DataArrival event of a **Winsock** control. When the event occurs, the code invokes the **GetData** method to retrieve the data and store it in a string variable. The data is then written into a **TextBox** control.

```
Private Sub Winsock1_DataArrival _
(ByVal bytesTotal As Long)
   Dim strData As String
   Winsock1.GetData strData, vbString
   Text1.Text = Text1.Text & strData
End Sub
```

Visual Basic Reference

Visual Studio 6.0

GetData Method

See Also Example Applies To

Returns a graphic from the **Clipboard** object. Doesn't support named arguments.

Syntax

object.**GetData (***format***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	Optional. A constant or value that specifies the **Clipboard** graphics format, as described in Settings. Parentheses must enclose the constant or value. If *format* is 0 or omitted, **GetData** automatically uses the appropriate 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.

If no graphic on the **Clipboard** object matches the expected format, nothing is returned. If only a color palette is present on the **Clipboard** object, a minimum size (1 x 1) DIB is created.

Visual Basic Reference

GetData Method Example

This example uses the **GetData** method to copy a bitmap from the **Clipboard** object to 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 ()
   Const CF_BITMAP = 2   ' Define bitmap format.
   Dim Msg   ' Declare variable.
   On Error Resume Next   ' Set up error handling.
   Msg = "Choose OK to load a bitmap onto the Clipboard."
   MsgBox Msg   ' Display message.
   Clipboard.Clear   ' Clear Clipboard.
   Clipboard.SetData LoadPicture("PAPER.BMP")  ' Get bitmap.
   If Err Then
      Msg = "Can't find the .bmp file."
      MsgBox Msg   ' Display error message.
      Exit Sub
   End If
   Msg = "A bitmap is now on the Clipboard. Choose OK to copy "
   Msg = Msg & "the bitmap from the Clipboard to the form "
   MsgBox Msg   ' Display message.
   Picture = Clipboard.GetData()   ' Copy from Clipboard.
   Msg = "Choose OK to clear the form."
   MsgBox Msg   ' Display message.
   Picture = LoadPicture()   ' Clear form.
End Sub
```

Visual Basic for Applications Reference

Visual Studio 6.0

GetDrive Method

See Also Example Applies To Specifics

Description

Returns a **Drive** object corresponding to the drive in a specified path.

Syntax

object.**GetDrive** *drivespec*

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

Part	Description
object	Required. Always the name of a **FileSystemObject**.
drivespec	Required. The *drivespec* argument can be a drive letter (c), a drive letter with a colon appended (c:), a drive letter with a colon and path separator appended (c:\), or any network share specification (\\computer2\share1).

Remarks

For network shares, a check is made to ensure that the share exists.

An error occurs if *drivespec* does not conform to one of the accepted forms or does not exist.

To call the **GetDrive** method on a normal path string, use the following sequence to get a string that is suitable for use as *drivespec*:

```
DriveSpec = GetDriveName(GetAbsolutePathName(Path))
```

Visual Basic for Applications Reference

Visual Studio 6.0

GetDriveName Method

See Also Example Applies To Specifics

Description

Returns a string containing the name of the drive for a specified path.

Syntax

object.**GetDriveName(***path***)**

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

Part	Description
object	Required. Always the name of a **FileSystemObject**.
path	Required. The path specification for the component whose drive name is to be returned.

Remarks

The **GetDriveName** method returns a zero-length string ("") if the drive can't be determined.

Note The **GetDriveName** method works only on the provided *path* string. It does not attempt to resolve the path, nor does it check for the existence of the specified path.

Visual Basic for Applications Reference

Visual Studio 6.0

GetExtensionName Method

See Also Example Applies To Specifics

Description

Returns a string containing the extension name for the last component in a path.

Syntax

object.**GetExtensionName(***path***)**

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

Part	Description
object	Required. Always the name of a **FileSystemObject**.
path	Required. The path specification for the component whose extension name is to be returned.

Remarks

For network drives, the root directory (\) is considered to be a component.

The **GetExtensionName** method returns a zero-length string ("") if no component matches the *path* argument.

Visual Basic for Applications Reference

Visual Studio 6.0

GetFile Method

See Also Example Applies To Specifics

Description

Returns a **File** object corresponding to the file in a specified path.

Syntax

object.**GetFile(***filespec***)**

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

Part	Description
object	Required. Always the name of a **FileSystemObject**.
filespec	Required. The *filespec* is the path (absolute or relative) to a specific file.

Remarks

An error occurs if the specified file does not exist.

Visual Basic for Applications Reference

Visual Studio 6.0

GetFileName Method

See Also Example Applies To Specifics

Description

Returns the last component of specified path that is not part of the drive specification.

Syntax

object.**GetFileName(***pathspec***)**

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

Part	Description
object	Required. Always the name of a **FileSystemObject**.
pathspec	Required. The path (absolute or relative) to a specific file.

Remarks

The **GetFileName** method returns a zero-length string ("") if *pathspec* does not end with the named component.

Note The **GetFileName** method works only on the provided path string. It does not attempt to resolve the path, nor does it check for the existence of the specified path.

Visual Basic: Windows Controls

Visual Studio 6.0

GetFirstVisible Method

See Also Example Applies To

Returns a reference to the first object visible in the internal area of a control.

Syntax

object.**GetFirstVisible()**

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

Remarks

A **ListView** control can contain more **ListItem** objects than can be seen in the internal area of the **ListView** control. You can use the reference returned by the **GetFirstVisible** method to determine the first visible **ListItem** object in List or Report view.

Visual Basic: Windows Controls

GetFirstVisible Method Example

This example populates a **ListView** control with the contents of the Publishers table in the Biblio.mdb database. When you click on the **CommandButton** control, the text of the first visible item is displayed. Click on the column headers to change the **SortKey** property and click the **CommandButton** again. To try the example, place a **ListView** 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 using the References command from the Tools menu. Run the example.

```
Private Sub Command1_Click()
   ' Create a ListItem variable and set the variable to the object
   ' returned by the GetFirstVisible method. Use the reference to
   ' display the text of the ListItem.
   Dim itmX As ListItem
   Set itmX = ListView1.GetFirstVisible
   MsgBox itmX.Text
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)

   ListView1.BorderStyle = ccFixedSingle ' Set BorderStyle property.

   ' 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))

      ' 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 the 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 = lvwReport   ' Set view to Report.
End Sub

Private Sub ListView1_ColumnClick(ByVal ColumnHeader As ColumnHeader)
   ListView1.SortKey = ColumnHeader.Index - 1
   ListView1.Sorted = True
End Sub
```

Visual Basic for Applications Reference

Visual Studio 6.0

GetFolder Method

See Also Example Applies To Specifics

Description

Returns a **Folder** object corresponding to the folder in a specified path.

Syntax

object.**GetFolder(***folderspec***)**

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

Part	Description
object	Required. Always the name of a **FileSystemObject**.
folderspec	Required. The *folderspec* is the path (absolute or relative) to a specific folder.

Remarks

An error occurs if the specified folder does not exist.

Visual Basic Reference

Visual Studio 6.0

GetFormat Method (ActiveX Controls)

See Also Example Applies To

Returns an integer indicating whether an item on the **Clipboard** object matches a specified format. Doesn't support named argument.

Syntax

object.**GetFormat (***format***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	Required. A value or constant that specifies the **Clipboard** object format, as described in Settings. Parentheses must enclose the constant or value.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFLink	&HBF00	DDE conversation information
vbCFText	1	Text
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.

The **GetFormat** method returns **True** if an item on the **Clipboard** object matches the specified format. Otherwise, it returns **False**.

For **vbCFDIB** and **vbCFBitmap** formats, whatever color palette is on the **Clipboard** is used when the graphic is displayed.

Visual Basic Reference

Visual Studio 6.0

GetFormat Method (DataObject Object)

See Also Example Applies To

Returns an boolean value indicating whether an item in the **DataObject** object matches a specified format. Doesn't support named arguments.

Syntax

object.**GetFormat** *format*

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

Part	Description
Object	Required. An object expression that evaluates to an object in the Applies To list.
Format	A constant or value that specifies the data format, 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 **GetFormat** method returns **True** if an item in the **DataObject** object matches the specified format. Otherwise, it returns **False**.

Visual Basic Reference

Visual Studio 6.0

GetFormat Method

See Also Example Applies To

Returns an integer indicating whether an item on the **Clipboard** object matches a specified format. Doesn't support named arguments.

Syntax

object.**GetFormat (***format***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	Required. A value or constant that specifies the **Clipboard** object format, as described in Settings. Parentheses must enclose the constant or value.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFLink	&HBF00	DDE conversation information
vbCFText	1	Text
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.

The **GetFormat** method returns **True** if an item on the **Clipboard** object matches the specified format. Otherwise, it returns **False**.

For **vbCFDIB** and **vbCFBitmap** formats, whatever color palette is on the **Clipboard** is used when the graphic is displayed.

Visual Basic Reference

GetFormat Method Example

This example uses the **GetFormat** method to determine the format of the data on the **Clipboard** object. 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 ()
   ' Define bitmap formats.
   Dim ClpFmt, Msg   ' Declare variables.
   On Error Resume Next   ' Set up error handling.
   If Clipboard.GetFormat(vbCFText) Then ClpFmt = ClpFmt + 1
   If Clipboard.GetFormat(vbCFBitmap) Then ClpFmt = ClpFmt + 2
   If Clipboard.GetFormat(vbCFDIB) Then ClpFmt = ClpFmt + 4
   If Clipboard.GetFormat(vbCFRTF) Then ClpFmt = ClpFmt + 8
   Select Case ClpFmt
      Case 1
         Msg = "The Clipboard contains only text."
      Case 2, 4, 6
         Msg = "The Clipboard contains only a bitmap."
      Case 3, 5, 7
         Msg = "The Clipboard contains text and a bitmap."
      Case 8, 9
         Msg = "The Clipboard contains only rich text."
      Case Else
         Msg = "There is nothing on the Clipboard."
   End Select
   MsgBox Msg   ' Display message.
End Sub
```

Visual Basic: Internet Control

Visual Studio 6.0

GetHeader Method

See Also Example Applies To

The **GetHeader** method is used to retrieve header text from an HTTP file.

Syntax

object.**GetHeader (***hdrName***)**

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
hdrName	Optional. A string that specifies the header to be retrieved.

Return Type

String

Remarks

If no header is named, all of the headers will be returned.

The table below shows some of the typical headers available.

Header	Description
`Date`	Returns the time and date of the document's transmission. The format of the returned data is `Wednesday, 27-April-96 19:34:15 GMT`.
`MIME-version`	Returns the MIME protocol version, which is currently 1.00.
`Server`	Returns the name of the server.
`Content-length`	Returns the length in bytes of the data.

`Content-type`	Returns the MIME `Content-type` of the data.
`Last-modified`	Returns the date and time of the document's last modification. The format of the returned data is `Wednesday, 27-April-96 19:34:15 GMT`.

Visual Basic: Internet Control

Visual Studio 6.0

GetChunk Method (Internet Transfer Control)

See Also Example Applies To

Retrieves data from in the StateChanged event. Use this method after invoking the **Execute** method as a **GET** operation.

Syntax

object.**GetChunk(** *size [,datatype]* **)**

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
size	Required. A long numeric expression that determines the size of the chunk to be retrieved.
datatype	Optional. An integer that specifies the data type of the retrieved chunk, as shown in Settings below.

Settings

The settings for *datatype* are:

Constant	Value	Description
icString	0	Default. Retrieves data as string.
icByteArray	1	Retrieves data as a byte array.

Return Type

Variant

Remarks

Use the **GetChunk** method in the StateChanged event. When the **State** property is **icResponseCompleted** (12), then use the **GetChunk** method to retrieve the buffer's contents.

Visual Basic: Internet Control

GetChunk Method, StateChanged Event Example

The example uses the **GetChunk** method in the StateChanged event to retrieve a chunk of data. The example uses a **Select Case** statement to determine what to do with every possible state. The example assumes a **TextBox** control named **txtData** exists on the form.

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
   ' Retrieve server response using the GetChunk
   ' method when State = 12. This example assumes the
   ' data is text.

   Select Case State
   ' ... Other cases not shown.

   Case icResponseReceived ' 12
      Dim vtData As Variant ' Data variable.
      Dim strData As String: strData = ""
      Dim bDone As Boolean: bDone = False

      ' Get first chunk.
      vtData = Inet1.GetChunk(1024, icString)
      DoEvents
      Do While Not bDone
         strData = strData & vtData
         DoEvents
         ' Get next chunk.
         vtData = Inet1.GetChunk(1024, icString)
         If Len(vtData) = 0 Then
            bDone = True
         End If
      Loop

      txtData.Text = strData
   End Select

End Sub
```

Visual Basic: RDO Data Control

Visual Studio 6.0

GetChunk Method (Remote Data)

See Also Example Applies To

Returns all or a portion of the contents of an **rdoColumn** object with a data type of **rdTypeLONGVARBINARY** or **rdTypeLONGVARCHAR**.

Syntax

varname = *object* **!** *column*.**GetChunk(***numbytes***)**

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

Part	Description
varname	The name of a Variant that receives the data from the **rdoColumn** object named by *column*.
object	An object expression that evaluates to an **rdoResultset** object containing the **rdoColumns** collection.
column	An object expression that evaluates to an **rdoColumn** object whose **ChunkRequired** property is **True**.
numbytes	A numeric expression that is the number of bytes you want to return.

Remarks

Chunk data columns are designed to store binary or text values that can range in size from a few characters to over 1.2GB and are stored in the database on successive data pages. In most cases, chunk data cannot be managed with a single operation so you must use the chunk methods to save and write data a piece at a time. If the **ChunkRequired** property is **True** for a column, you should use the **GetChunk** and **AppendChunk** methods to manipulate column data. The **BindThreshold** property determines the largest size block that is automatically bound and precludes the need to use the chunk methods.

If the **ChunkRequired** property is **True** for a column, you must use the **GetChunk** method to retrieve the data. The **GetChunk** method moves a portion of the data from a chunk column to a variable. The total number of bytes in the column is determined by executing the **ColumnSize** method.

The **GetChunk** method is used iteratively, copying column data to a variable, one segment or chunk at a time. The chunk size is set by ***numbytes***. The starting point of the copy operation is initially 0, which causes data to be copied from the first byte of the column being read. Subsequent calls to **GetChunk** get data from the first position after the previously read chunk.

The bytes returned by **GetChunk** are assigned to *varname*. Due to memory requirements for the returned data and temporary storage, *numbytes* might be limited, but with 32-bit systems this limitation is over 1.2GB, or more practically the memory and disk capacity of your virtual memory system.

If *numbytes* is greater than the number of bytes in the column, the actual number of bytes in the column is returned. After assigning the results of **GetChunk** to a **Variant** variable, you can use the **Len** function to determine the number of bytes returned.

Use the **AppendChunk** method to write successive blocks of data to the column and **GetChunk** to extract data from the column. Certain operations (copying, for example) involve temporary strings. If string space is limited, you may need to work with smaller segments of a *chunk* column instead of the entire column.

Use the **BindThreshold** property to specify the largest column size that will be automatically bound.

Note Because the size of a chunk data column can exceed 1.2GB, you should assign the value returned by the **GetChunk** method to a variable large enough to store the data returned based on the size returned by the **ColumnSize** method.

Visual Basic: RDO Data Control

AppendChunk, GetChunk Method Example

This example illustrates use of the **AppendChunk** and **GetChunk** methods to write page-based binary large object (BLOB) data to a remote data source. The code expects a table with a char, text, and image field named *Chunks*. To create this table, submit the following as an action query against your test database:

```
CREATE TABLE Chunks (ID integer identity NOT NULL, PName char(10) NULL,
Description TEXT NULL,
Photo IMAGE NULL)
CREATE UNIQUE INDEX ChunkIDIndex on Chunks(ID)
```

Once the table is created, you will need to locate one or more .BMP or other suitable graphics images that can be loaded by the **PictureBox** control.

```
'
Option Explicit
Dim en As rdoEnvironment
Dim Qd As rdoQuery
Dim Cn As rdoConnection
Dim Rs As rdoResultset
Dim SQL As String
Dim DataFile As Integer, Fl As Long, Chunks As Integer
Dim Fragment As Integer, Chunk() As Byte, I As Integer
Const ChunkSize As Integer = 16384

Private Sub Form_Load()
Set en = rdoEnvironments(0)
Set Cn = en.OpenConnection(dsname:="", _
   Connect:="UID=;PWD=;DATABASE=WorkDB;" _
   & "Driver={SQL Server};SERVER=Betav486", _
   prompt:=rdDriverNoPrompt)
Set Qd = Cn.CreateQuery("TestChunk", "Select * from
   Chunks Where PName = ?")
End Sub

Private Sub LoadFromFile_Click()
'
'  Locates a file and sets the Filename to this file.
'
With CommonDialog1
   .Filter = "Pictures(*.bmp;*.ico)|*.bmp;*.ico"
   .ShowOpen
   FileName = .FileName
End With
End Sub

Private Sub ReadFromDB_Click()
If Len(NameWanted) = 0 Then _
   NameWanted = InputBox("Enter name wanted", "Animal")
   Qd(0) = NameWanted
   Set Rs = Qd.OpenResultset(rdOpenKeyset, _
   rdConcurRowver)
```

```
If Rs Is Nothing Or Rs.Updatable = False Then
   MsgBox "Cant open or write to result set"
   Exit Sub
End If
If Rs.EOF Then
   MsgBox "Cant find picture by that name"
   Exit Sub
End If
Description = Rs!Description
DataFile = 1
Open "pictemp" For Binary Access Write As DataFile
Fl = Rs!Photo.ColumnSize
Chunks = Fl \ ChunkSize
Fragment = Fl Mod ChunkSize
ReDim Chunk(Fragment)
Chunk() = Rs!Photo.GetChunk(Fragment)
Put DataFile, , Chunk()
For I = 1 To Chunks
   ReDim Buffer(ChunkSize)
   Chunk() = Rs!Photo.GetChunk(ChunkSize)
   Put DataFile, , Chunk()
Next I
Close DataFile
FileName = "pictemp"
End Sub

Private Sub SaveToDB_Click()
If Len(NameWanted) = 0 Then _
   NameWanted = InputBox("Enter name for this" _
   & " picture", "Animal")
   Qd(0) = NameWanted
   Set Rs = Qd.OpenResultset(rdOpenKeyset, _
   rdConcurRowver)
If Rs Is Nothing Or Rs.Updatable = False Then
   MsgBox "Cant open or write to result set"
   Exit Sub
End If
If Rs.EOF Then
   Rs.AddNew
   Rs!PName = NameWanted
If Description = "" Then  _
   Description = InputBox("Describe the picture", _
   "Dont care")
   'Rs!Description = Description
Else
   Rs.Edit
End If
DataFile = 1
Open FileName For Binary Access Read As DataFile
Fl = LOF(DataFile)    ' Length of data in file
If Fl = 0 Then Close DataFile: Exit Sub
Chunks = Fl \ ChunkSize
Fragment = Fl Mod ChunkSize
Rs!Photo.AppendChunk Null
ReDim Chunk(Fragment)
Get DataFile, , Chunk()
Rs!Photo.AppendChunk Chunk()
ReDim Chunk(ChunkSize)
For I = 1 To Chunks
   Get DataFile, , Chunk()
   Rs!Photo.AppendChunk Chunk()
```

```
Next I
Close DataFile
Rs.Update
End Sub

Private Sub FileName_Change()
Picture1.Picture = LoadPicture(FileName)
End Sub
```

Visual Basic: RichTextBox Control

Visual Studio 6.0

GetLineFromChar Method

See Also Example Applies To

Returns the number of the line containing a specified character position in a **RichTextBox** control.

Syntax

object.**GetLineFromChar**(*charpos*)

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
charpos	Required. A long integer that specifies the index of the character whose line you want to identify. The index of the first character in the **RichTextBox** control is 0.

Remarks

You use the **GetLineFromChar** method to find out which line in the text of a **RichTextBox** control contains a certain character position in the text. You might need to do this because the number of characters in each line of text can vary, making it very difficult to find out which line in the text contains a particular character, identified by its position in the text.

Visual Basic: RichTextBox Control

GetLineFromChar Method Example

This example finds a string in a **RichTextBox** control based on a word entered in a **TextBox** control. After it finds the specified string, it displays a message box that shows the number of the line containing the specified word. To try this example, put a **RichTextBox** control, a **CommandButton** control and a **TextBox** control on a form. Load a file into the **RichTextBox,** and paste this code into the General Declarations section of the form. Then run the example, enter a word in the **TextBox**, and click the **CommandButton**.

```
Private Sub Command1_Click()
   Dim FoundPos As Integer
   Dim FoundLine As Integer
   ' Find the text specified in the TextBox control.
   FoundPos = RichTextBox1.Find(Text1.Text, , , rtfWholeWord)

   ' Show message based on whether the text was found or not.

   If FoundPos <> -1 Then
      ' Returns number of line containing found text.
      FoundLine = RichTextBox1.GetLineFromChar(FoundPos)
      MsgBox "Word found on line " & CStr(FoundLine)
   Else
      MsgBox "Word not found."
   End If
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

GetNumTicks Method

See Also Example Applies To

Returns the number of ticks between the **Min** and **Max** properties of the **Slider** control.

Syntax

object.**GetNumTicks**

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

Remarks

To change the number of ticks, reset the **Min** or **Max** properties or the **TickFrequency** property.

Visual Basic: Windows Controls

GetNumTicks Method Example

This example displays the current number of ticks on a **Slider** control, then increments the **Max** property by 10. To try this example, place a **Slider** control onto a form and paste the code into the form's Declarations section. Run the example, and click the **Slider** control to get the number of ticks. Every click on the control increases the ticks.

```
Private Sub Slider1_Click()
   MsgBox Slider1.GetNumTicks
   Slider1.Max = Slider1.Max + 10
End Sub
```

Visual Basic for Applications Reference

Visual Studio 6.0

GetParentFolderName Method

See Also Example Applies To Specifics

Description

Returns a string containing the name of the parent folder of the last component in a specified path.

Syntax

object.**GetParentFolderName(***path***)**

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

Part	Description
object	Required. Always the name of a **FileSystemObject**.
path	Required. The path specification for the component whose parent folder name is to be returned.

Remarks

The **GetParentFolderName** method returns a zero-length string ("") if there is no parent folder for the component specified in the *path* argument.

Note The **GetParentFolderName** method works only on the provided *path* string. It does not attempt to resolve the path, nor does it check for the existence of the specified path.

Visual Basic: RDO Data Control

Visual Studio 6.0

GetRows Method (Remote Data)

See Also Example Applies To

Retrieves multiple rows of an **rdoResultset** into an array.

Syntax

array = *object*.**GetRows (***rows***)**

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

Part	Description
array	The name of a Variant type variable to store the returned data.
object	An object expression that evaluates to an object in the Applies To list.
rows	A Long value indicating the number of rows to retrieve.

Remarks

Use the **GetRows** method to copy one or more entire rows from an **rdoResultset** into a two-dimensional array. The first array subscript identifies the column and the second identifies the row number, as follows:

```
avarRows(intColumn)(intRow)
```

To get the first column value in the second row returned, use the following:

```
col1 = avarRows(0,1)
```

To get the second column value in the first row, use the following:

```
col2 = avarRows(1,0)
```

If more rows are requested than are available, only the available rows are returned. Use **Ubound** to determine how many rows are actually fetched, as the array is resized based on the number of rows returned. For example, if you return the results into a **Variant** called varA, you could determine how many rows were actually returned by using:

```
numReturned = Ubound(varA,2) + 1
```

The + 1 is used because the first data returned is in the 0^{th} element of the array. The number of rows that can be fetched is constrained by available memory and should be chosen to suit your application dont expect to use **GetRows** to bring your entire table or result set into an array if it is a large table.

GetRows does not return data from columns whose **ChunkRequired** property is **True** a variant value containing an ODBC S-code is returned in these columns instead.

After a call to **GetRows**, the current row is positioned at the next unread row. That is, **GetRows** is equivalent to using the **Move** (*rows*) method.

If you are trying to fetch all the rows using multiple **GetRows** calls, use the **EOF** property to determine if there are rows available. **GetRows** returns less than the number requested either at the end of the **rdoResultset**, or if it cannot fetch a row in the range requested. For example, if a fifth row cannot be retrieved in a group of ten rows that youre trying to fetch, **GetRows** returns four rows and leaves currency on the row that caused the problem. It will not generate a run-time error.

The **GetRows** method fetches data from the ODBC buffers based on the **RowsetSize** property. RDO proceeds to fetch from the current row toward the end of the result set returning as many rows as you requested. As the current rowset is exhausted, RDO issues another **SQLExtendedFetch** function call to fetch subsequent rowsets from the database. This technique applies to all types of cursors.

Visual Basic: RDO Data Control

GetRows Method Example

This example illustrates use of the **GetRows** method to fetch rows from an **rdoResultset** into a variant array. The code opens a connection to a remote data source and creates an **rdoQuery** object that requires a single parameter. The `GetRowsNow` procedure executes the query with a user-supplied parameter and uses **GetRows** to fetch the rows from the result set.

```
Option Explicit
Dim er As rdoError
Dim cn As New rdoConnection
Dim qy As New rdoQuery
Dim rs As rdoResultset
Dim RowBuf As Variant
Dim RowsReturned As Integer
Dim i As Integer
Dim Ans As Integer

Private Sub GetRowsNow_Click()
qy(0) = StateWanted
rs.Requery

Do Until rs.EOF
   List1.Clear
   RowBuf = rs.GetRows(5)      'Get the next 5 rows
   RowsReturned = UBound(RowBuf, 2) + 1
   For i = 0 To RowsReturned - 1
      List1.AddItem RowBuf(0, i) & ":" & RowBuf(1, i)
   Next i
   Ans = MsgBox("Press Ok to see next 5 rows " _
      &" or Cancel to quit", vbOKCancel)
   If Ans = vbOK Then Else Exit Sub
Loop
End Sub

Private Sub Form_Load()
cn.CursorDriver = rdUseOdbc
cn.Connect = "uid=;pwd=;server=SEQUEL;" _
   driver={SQL Server};database=pubs;dsn='';"
cn.EstablishConnection
With qy
   .Name = "GetRowsQuery"
   .SQL = "Select * from Titles T, Publishers P " _
   & " Where T.Pub_ID = P.Pub_ID " _
   & " and P.State = ?"
   .RowsetSize = 1
   Set .ActiveConnection = cn
   .rdoParameters(0) = "CA"
   Set rs = .OpenResultset(rdOpenKeyset, _
      rdConcurRowver)
End With
End Sub
```

Visual Studio 6.0

Visual Basic: MSChart Control

GetSelectedPart Method

See Also Example Applies To

Identifies the currently selected chart element.

Syntax

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

The **GetSelectedPart** 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 element. Valid constants are VtChPartType.
index1	Integer. If element 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 element refers to an axis or axis label, this argument identifies the axis type with a VtChAxisId constant.
Index2	Integer. If element refers to a data point, this argument specifies which data point in the series identified by index1.
index3	Integer. If element 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 element is not an axis label, the argument is unused.
index4	Integer. This argument is unused at this time.

Visual Basic Extensibility Reference

Visual Studio 6.0

GetSelection Method

See Also Example Applies To Specifics

Returns the selection in a code pane.

Syntax

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

The **GetSelection** 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 that returns a value specifying the first line of the selection in the code pane.
startcol	Required. A **Long** that returns a value specifying the first column of the selection in the code pane.
endline	Required. A **Long** that returns a value specifying the last line of the selection in the code pane.
endcol	Required. A **Long** that returns a value specifying the last column of the selection in the code pane.

Remarks

When you use the **GetSelection** method, information is returned in output arguments. As a result, you must pass in variables because the variables will be modified to contain the information when returned.

Visual Basic Extensibility Reference

GetSelection Method Example

The following example returns the locations of the starting and ending points of the current selection in `CodePanes(1)`. The last line in the example uses the **GetSelection** method to place the four values in the four variables.

```
Dim m As Long
Dim n As Long
Dim x As Long
Dim y As Long
Application.VBE.CodePanes(1).GetSelection m, n, x, y
```

Visual Basic for Applications Reference

Visual Studio 6.0

GetSpecialFolder Method

See Also Example Applies To Specifics

Description

Returns the special folder specified.

Syntax

object.**GetSpecialFolder(***folderspec***)**

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

Part	Description
object	Required. Always the name of a **FileSystemObject**.
folderspec	Required. The name of the special folder to be returned. Can be any of the constants shown in the Settings section.

Settings

The *folderspec* argument can have any of the following values:

Constant	Value	Description
WindowsFolder	0	The Windows folder contains files installed by the Windows operating system.
SystemFolder	1	The System folder contains libraries, fonts, and device drivers.
TemporaryFolder	2	The Temp folder is used to store temporary files. Its path is found in the TMP environment variable.

Visual Basic for Applications Reference

Visual Studio 6.0

GetTempName Method

See Also Example Applies To Specifics

Description

Returns a randomly generated temporary file or folder name that is useful for performing operations that require a temporary file or folder.

Syntax

object.**GetTempName**

The optional *object* is always the name of a **FileSystemObject**.

Remarks

The **GetTempName** method does not create a file. It provides only a temporary file name that can be used with **CreateTextFile** to create a file.

Visual Basic Reference

Visual Studio 6.0

GetText Method

See Also Example Applies To

Returns a text string from the **Clipboard** object. Doesn't support named arguments.

Syntax

object.**GetText (***format***)**

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	Optional. A value or constant that specifies the **Clipboard** object format, as described in Settings. Parentheses must enclose the constant or value.

Settings

The settings for *format* are:

Constant	Value	Description
vbCFLink	&HBF00	DDE conversation information
vbCFText	1	(Default) Text
vbCFRTF	&HBF01	Rich Text Format (.rtf file)

Remarks

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

If no text string on the **Clipboard** object matches the expected format, a zero-length string ("") is returned.

Visual Basic Reference

GetText Method Example

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

```
Private Sub Form_Click ()
   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(vbCFText) 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(vbCFText)   ' 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: Windows Controls

Visual Studio 6.0

GetVisibleCount Method

See Also Example Applies To

Returns the number of **Node** objects that fit in the internal area of a **TreeView** control.

Syntax

object.**GetVisibleCount**

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

Remarks

The number of **Node** objects is determined by how many lines can fit in a window. The total number of lines possible is determined by the height of the control and the **Size** property of the **Font** object. The count includes the partially visible item at the bottom of the list.

You can use the **GetVisibleCount** property to make sure that a minimum number of lines are visible so the user can accurately assess a hierarchy. If the minimum number of lines is not visible, you can reset the size of the **TreeView** using the **Height** property.

If a particular **Node** object must be visible, use the **EnsureVisible** method to scroll and expand the **TreeView** control.

Visual Basic: Windows Controls

GetVisibleCount Method Example

This example adds several **Node** objects to a **TreeView** control. When you click the form, the code uses the **GetVisibleCount** method to check how many lines are visible, and then enlarges the control to show all the objects. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and click the form to enlarge the control.

```
Private Sub Form_Load()
   Dim nodX As Node
   Dim i as Integer
   TreeView1.BorderStyle = 1 ' Show border.
   For i = 1 to 20
      Set nodX = TreeView1.Nodes.Add(,,,"Node " & CStr(i))
   Next I
   TreeView1.Height = 1500 ' TreeView is short, for comparison's sake.
End Sub

Private Sub Form_Click()
   While Treeview1.GetVisibleCount < 20
      ' Make the treeview larger.
      TreeView1.Height = TreeView1.Height + TreeView1.Font.Size
   Wend
End Sub
```

Visual Basic Reference

Visual Studio 6.0

GoBack Method

See Also Example Applies To

Execute a hyperlink jump back in the history list.

Syntax

object.**GoBack**

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

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

Remarks

If the object is in a container that supports OLE hyperlinking, then the container will jump to the location that is back in the history list. If the object is in a container that does not support OLE hyperlinking, then this method will raise an error.

Visual Basic Reference

Visual Studio 6.0

GoForward Method

See Also Example Applies To

Execute a hyperlink jump forward in the history list.

Syntax

object.**GoForward**

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

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

Remarks

If the object is in a container that supports OLE hyperlinking, then the container will jump to the location that is forward in the history list. If the object is in a container that does not support OLE hyperlinking, then this method will raise an error.