Visual Basic Reference

Visual Studio 6.0

Handle Property

See Also Example Applies To

Returns a handle to the graphic contained within a **Picture** object.

Syntax

object.**Handle**

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

Return Value

The value returned by the **Handle** property depends on the current setting of the **Type** property as shown in the following table:

Type Property	Return Value
1 (Bitmap)	An HBITMAP handle.
2 (Metafile)	An HMETAFILE handle.
3 (Icon)	An HICON or an HCURSOR handle.
4 (Enhanced Metafile)	An HENHMETAFILE handle.

Remarks

The **Handle** property is useful when you need to pass a handle to a graphic as part of a call to a function in a dynamic-link library (DLL) or the Windows API.

Visual Basic: MSComm Control

Visual Studio 6.0

Handshaking Property

See Also Example Applies To

Sets and returns the hardware handshaking protocol.

Syntax

object.**Handshaking** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer expression specifying the handshaking protocol, as described in Settings.

Settings

The settings for *value* are:

Setting	Value	Description
comNone	0	(Default) No handshaking.
comXOnXOff	1	XON/XOFF handshaking.
comRTS	2	RTS/CTS (Request To Send/Clear To Send) handshaking.
comRTSXOnXOff	3	Both Request To Send and XON/XOFF handshaking.

Remarks

Handshaking refers to the internal communications protocol by which data is transferred from the hardware port to the receive buffer. When a character of data arrives at the serial port, the communications device has to move it into the receive buffer so that your program can read it. If there is no receive buffer and your program is expected to read every character directly from the hardware, you will probably lose data because the characters can arrive very quickly.

A **handshaking** protocol insures data is not lost due to a buffer overrun, where data arrives at the port too quickly for the communications device to move the data into the receive buffer.

Data Type

Integer

Visual Basic Reference

Visual Studio 6.0

HasDC Property

See Also Example Applies To

Returns or sets a value that determines whether a unique display context (or hDC) is allocated to a control.

Syntax

object.**HasDC** [= *boolean*]

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

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies whether an hDC is allocated to a control.

Settings

The settings for *boolean* are:

Setting	**Description**
True	(Default) An hDC is allocated to the control.
False	No hDC is allocated to the control.

Remarks

If you set the **HasDC** property to **False**, you should copy the hDC of the control only into a local variable. Calling APIs with an hDC obtained outside of the scope of an event can cause crashes or other unpredictable results.

For windowless **UserControls**, the **HasDC** property will only take effect if the UserControl is created with a window (i.e., it's in a container that doesn't support windowless activation, such as Visual Basic version 4.0 or Internet Explorer version 3.0). Windowless **UserControls** that are activated windowless never have their own hDC, and therefore ignore the value of their **HasDC** property.

Visual Basic Extensibility Reference

Visual Studio 6.0

HasOpenDesigner Property

See Also Example Applies To Specifics

Returns a Boolean value indicating whether or not the **VBComponent** object has an open designer. Read-only.

Return Values

The **HasOpenDesigner** property returns these values:

Value	Description
True	The **VBComponent** object has an open **Design** window.
False	The **VBComponent** object doesn't have an open **Design** window.

Visual Basic Extensibility Reference

HasOpenDesigner Property Example

The following example uses the **HasOpenDesigner** property to return whether or not the specified component, in this case a form, of a particular project has an open designer.

```
Debug.Print Application.VBE.VBProjects(1).VBComponents(1).HasOpenDesigner
```

Visual Basic: RDO Data Control

Visual Studio 6.0

hDbc Property (Remote Data)

See Also Example Applies To

Returns a value corresponding to the ODBC connection handle.

Syntax

object.**hDbc**

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

Return Values

The **hDbc** property returns a Long value containing the ODBC connection handle created by the ODBC driver manager corresponding to the specified **rdoConnection** object.

Remarks

This handle can be used to execute ODBC functions that require an ODBC **hDbc** connection handle.

Note While it is possible to execute ODBC API functions using the ODBC **hEnv, hDbc**, and **hStmt** handles, it is recommended that you do so with caution. Improper use of arbitrary ODBC API functions using these handles can result in unpredictable behavior. You should not attempt to save this handle in a variable for use at a later time as the value is subject to change.

If your application requires access to special ODBC connection option settings, these should be set or retrieved using the **hDbc** property *before* the connection is established. Resetting ODBC settings of any kind after the connection is established can result in unpredictable behavior.

Visual Basic: RDO Data Control

hDbc Property Example

The following example illustrates use of the hDbc property when executing an ODBC API function. In this case, the application sets a connection option that changes how transactions are isolated.

```
Option Explicit
Dim en As rdoEnvironment
Dim cn As rdoConnection
Dim rc As Integer

'Declare Function SQLSetConnectOption Lib "odbc32.dll" (ByVal hdbc&, ByVal fOption%, ByVal vParam As Any) As Integer
'
'Transaction isolation option masks
'
 Const SQL_TXN_ISOLATION As Long = 108
 Const SQL_TXN_READ_UNCOMMITTED As Long = &H1&
 Const SQL_TXN_READ_COMMITTED As Long = &H2&
 Const SQL_TXN_REPEATABLE_READ As Long = &H4&
 Const SQL_TXN_SERIALIZABLE As Long = &H8&
 Const SQL_TXN_VERSIONING As Long = &H10&

Private Sub Form_Load()
Set en = rdoEngine.rdoEnvironments(0)

Set cn = en.OpenConnection(dsName:="WorkDB", _
    Prompt:=rdDriverNoPrompt, _
    Connect:="Uid=;pwd=;database=workdb")

rc = SQLSetConnectOption(cn.hDbc, SQL_TXN_ISOLATION, SQL_TXN_READ_UNCOMMITTED)

Debug.Print rc

End Sub
```

Visual Basic Reference

Visual Studio 6.0

hDC Property

See Also Example Applies To

Returns a handle provided by the Microsoft Windows operating environment to the device context of an object.

Syntax

object.**hDC**

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

Remarks

This property is a Windows operating environment device context handle. The Windows operating environment manages the system display by assigning a device context for the **Printer** object and for each form and **PictureBox** control in your application. You can use the **hDC** property to refer to the handle for an object's device context. This provides a value to pass to Windows API calls.

With a **CommonDialog** control, this property returns a device context for the printer selected in the Print dialog box when the **cdlReturnDC** flag is set or an information context when the **cdlReturnIC** flag is set.

Note The value of the **hDC** property can change while a program is running, so don't store the value in a variable; instead, use the **hDC** property each time you need it.

The **AutoRedraw** property can cause the **hDC** property setting to change. If **AutoRedraw** is set to **True** for a form or **PictureBox** container, **hDC** acts as a handle to the device context of the persistent graphic (equivalent to the **Image** property). When **AutoRedraw** is **False**, **hDC** is the actual **hDC** value of the Form window or the **PictureBox** container. The **hDC** property setting may change while the program is running regardless of the **AutoRedraw** setting.

If the **HasDC** property is set to **False**, a new device context will be created by the system and the value of the **hDC** property will change each time it is called.

Visual Basic Reference

hDC Property Example

This example draws a triangle and then uses a Microsoft Windows function to fill it with color. To try this example, create a new module using the Add Module command on the Project menu. Paste the **Declare** statement into the Declarations section of the new module, being sure that the statement is on one line with no break or wordwrap. Then paste the **Sub** procedure into the Declarations section of a form. Press F5 and click the form.

```
' Declaration of a Windows routine. This statement is
' for the module.
Declare Sub FloodFill Lib "GDI32" Alias "FloodFill" _
 (ByVal hDC As Long, ByVal X As Long, ByVal Y As _
 Long, ByVal crColor As Long) As Long
' Place the following code in the form.
Private Sub Form_Click ()
   ScaleMode = vbPixels     ' Windows draws in pixels.
   ForeColor = vbBlack     ' Set draw line to black.
   Line (100, 50)-(300, 50) ' Draw a triangle.
   Line -(200, 200)
   Line -(100, 50)
   FillStyle = vbFSSolid  ' Set FillStyle to solid.
   FillColor = RGB(128, 128, 255)   ' Set FillColor.
 ' Call Windows API to fill.
   FloodFill hDC, 200, 100, ForeColor
End Sub
```

Visual Basic Reference

Visual Studio 6.0

hDC Property (ActiveX Controls)

See Also Example Applies To

Returns a handle provided by the Microsoft Windows operating environment to the device context of an object.

Syntax

object.**hDC**

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

Remarks

This property is a Windows operating environment device context handle. The Windows operating environment manages the system display by assigning a device context for the **Printer** object and for each form and **PictureBox** control in your application. You can use the **hDC** property to refer to the handle for an object's device context. This provides a value to pass to Windows API calls.

With a **CommonDialog** control, this property returns a device context for the printer selected in the Print dialog box when the **cdlReturnDC** flag is set or an information context when the **cdlReturnIC** flag is set.

Note The value of the **hDC** property can change while a program is running, so don't store the value in a variable; instead, use the **hDC** property each time you need it.

The **AutoRedraw** property can cause the **hDC** property setting to change. If **AutoRedraw** is set to **True** for a form or **PictureBox** container, **hDC** acts as a handle to the device context of the persistent graphic (equivalent to the **Image** property). When **AutoRedraw** is **False**, **hDC** is the actual **hDC** value of the Form window or the **PictureBox** container. The **hDC** property setting may change while the program is running regardless of the **AutoRedraw** setting.

Visual Basic: DataGrid Control

Visual Studio 6.0

HeadFont Property

See Also Example Applies To

Returns or sets a value indicating the font used in column headers in a **DataGrid** control.

Syntax

object.**Type** [= *value*]

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

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
value	An object expression that evaluates to a **Font** object. The default value is the column's current font set to bold.

Remarks

Changing the **HeadFont** property may resize the headers to accommodate the new font.

Visual Basic: DataGrid Control

Visual Studio 6.0

HeadLines Property

See Also Example Applies To

Returns or sets a value indicating the number of lines of text displayed in the column headers of a **DataGrid** control.

Syntax

object.**HeadLines** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A from 0 to 10. The default value is 1, which causes the control to display the names of underlying fields for each column in the header. A setting of 0 removes the headings.

Remarks

The **HeadLines** property can be used to display more than one line of text in the column headers of the **DataGrid** control.

Visual Basic: DataGrid Control

HeadLines Property Example

This example checks the value of a check box to determine whether or not to display headings in the grid.

```
Private Sub Check1_Click ()
   If Check1.Value = vbChecked Then
      DataGrid1.HeadLines = 2   ' If checked,two lines in
                               ' column headings.
   Else
      DataGrid1.HeadLines = 0   ' No headings.
   End If
End Sub
```

Visual Basic Extensibility Reference

Visual Studio 6.0

Height Property (VBA Add-In Object Model)

See Also Example Applies To Specifics

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

Remarks

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

Visual Basic Extensibility Reference

Height, Width Properties Example

The following example uses the **Height** and **Width** properties to return the height and width of the specified window, in twips. These property settings change after a window is linked or docked because then they refer to the **Window** object to which the original window is linked or docked.

```
Debug.Print Application.VBE.Windows(9).Height
Debug.Print Application.VBE.Windows(9).Width
```

Visual Basic Reference

Visual Studio 6.0

Height, Width Properties

See Also Example Applies To

Return or set the dimensions of an object or the width of the **Columns** object of a **DataGrid** control. For the **Printer** and **Screen** objects, not available at design time.

Syntax

```
object.Height [= number]

object.Width [= number]
```

The **Height** and **Width** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression specifying the dimensions of an object, as described in Settings.

Settings

Measurements are calculated as follows:

- **Form** the external height and width of the form, including the borders and title bar.
- **Control** measured from the center of the control's border so that controls with different border widths align correctly. These properties use the scale units of a control's container.
- **Printer** object the physical dimensions of the paper set up for the printing device; not available at design time. If set at run time, values in these properties are used instead of the setting of the **PaperSize** property.
- **Screen** object the height and width of the screen; not available at design time and read-only at run time.
- **Picture** object the height and width of the picture in HiMetric units.

Remarks

For **Form**, **Printer**, and **Screen** objects, these properties are always measured in twips. For a form or control, the values for these properties change as the object is sized by the user or by your code. Maximum limits of these properties for all objects are system-dependent.

If you set the **Height** and **Width** properties for a printer driver that doesn't allow these properties to be set, no error occurs and the size of the paper remains as it was. If you set **Height** and **Width** for a printer driver that allows only certain values to be specified, no error occurs and the property is set to whatever the driver allows. For example, you could set **Height** to 150 and the driver would set it to 144.

Use the **Height**, **Width**, **Left**, and **Top** properties for operations or calculations based on an object's total area, such as sizing or moving the object. Use the **ScaleLeft**, **ScaleTop**, **ScaleHeight**, and **ScaleWidth** properties for operations or calculations based on an object's internal area, such as drawing or moving objects within another object.

Note The **Height** property can't be changed for the **DriveListBox** control or for the **ComboBox** control, whose **Style** property setting is 0 (Dropdown Combo) or 2 (Dropdown List).

For the **Columns** object of the **DataGrid** control, **Width** is specified in the unit of measure of the object that contains the **DataGrid**. The default value for **Width** is the value of the **DefColWidth** property of **DataGrid**.

For the **Picture** object, use the **ScaleX** and **ScaleY** methods to convert HiMetric units into the scale you need.

Visual Basic Reference

Height, Width Properties Example

This example sets the size of a form to 75 percent of screen size and centers the form when it is loaded. To try this example, paste the code into the Declarations section of a form. Then press F5 and click the form.

```
Private Sub Form_Click ()
   Width = Screen.Width * .75   ' Set width of form.
   Height = Screen.Height * .75   ' Set height of form.
   Left = (Screen.Width - Width) / 2   ' Center form horizontally.
   Top = (Screen.Height - Height) / 2   ' Center form vertically.
End Sub
```

Visual Basic Reference

Visual Studio 6.0

Height, Width Properties (ActiveX Controls)

See Also Example Applies To

Return or set the dimensions of an object or the width of the **Columns** object of a **DataGrid** control. For the **Printer** and **Screen** objects, not available at design time.

Syntax

object.**Height** [= *number*]

object.**Width** [= *number*]

The **Height** and **Width** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	A numeric expression specifying the dimensions of an object, as described in Settings.

Settings

Measurements are calculated as follows:

- **Form** the external height and width of the form, including the borders and title bar.
- **Control** measured from the center of the control's border so that controls with different border widths align correctly. These properties use the scale units of a control's container.
- **Printer** object the physical dimensions of the paper set up for the printing device; not available at design time. If set at run time, values in these properties are used instead of the setting of the **PaperSize** property.
- **Screen** object the height and width of the screen; not available at design time and read-only at run time.
- **Picture** object the height and width of the picture in HiMetric units.

Remarks

For **Form**, **Printer**, and **Screen** objects, these properties are always measured in twips. For a form or control, the values for these properties change as the object is sized by the user or by your code. Maximum limits of these properties for all objects are system-dependent.

If you set the **Height** and **Width** properties for a printer driver that doesn't allow these properties to be set, no error occurs and the size of the paper remains as it was. If you set **Height** and **Width** for a printer driver that allows only certain values to be specified, no error occurs and the property is set to whatever the driver allows. For example, you could set **Height** to 150 and the driver would set it to 144.

Use the **Height**, **Width**, **Left**, and **Top** properties for operations or calculations based on an object's total area, such as sizing or moving the object. Use the **ScaleLeft**, **ScaleTop**, **ScaleHeight**, and **ScaleWidth** properties for operations or calculations based on an object's internal area, such as drawing or moving objects within another object.

Note The **Height** property can't be changed for the **DriveListBox** control or for the **ComboBox** control, whose **Style** property setting is 0 (Dropdown Combo) or 2 (Dropdown List).

For the **Columns** object of the **DataGrid** control, **Width** is specified in the unit of measure of the object that contains the **DataGrid**. The default value for **Width** is the value of the **DefColWidth** property of **DataGrid**.

For the **Picture** object, use the **ScaleX** and **ScaleY** methods to convert HiMetric units into the scale you need.

Visual Basic: PictureClip Control

Visual Studio 6.0

Height, Width Properties (PictureClip Control)

See Also Example Applies To

Return the height and width (in pixels) of a bitmap contained in the control. These properties are not available at design time and are read-only at run time.

Syntax

[*form*.]*PictureClip*.**Height**

[*form*.]*PictureClip*.**Width**

Remarks

These properties are only valid when the control contains a bitmap.

You can load a bitmap into a **PictureClip** control at design time using the Properties sheet. In Visual Basic, you can also load a bitmap into a **PictureClip** control at run time by using the **LoadPicture** function.

Data Type

Integer

Visual Basic: CommonDialog Control

Visual Studio 6.0

HelpCommand Property

See Also Example Applies To

Returns or sets the type of online Help requested.

Syntax

object.**HelpCommand** [= *value*]

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

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

Settings

The settings for *value* are:

Constant	Value	Description
cdlHelpCommand	&H102&	Executes a Help macro.
cdlHelpContents	&H3&	Displays the Help contents topic as defined by the Contents option in the [OPTION] section of the .hpj file. See Remarks below for information on Help files created with Microsoft Help Workshop 4.0X.
cdlHelpContext	&H1&	Displays Help for a particular context. When using this setting, you must also specify a context using the **HelpContext** property.
CdlHelpContextPopup	&H8&	Displays in a pop-up window a particular Help topic identified by a context number defined in the [MAP] section of the .hpj file.
cdlHelpForceFile	&H9&	Ensures WinHelp displays the correct Help file. If the correct Help file is currently displayed, no action occurs. If the incorrect Help file is displayed, WinHelp opens the correct file.
cdlHelpHelpOnHelp	&H4&	Displays Help for using the Help application itself.

cdlHelpIndex	&H3&	Displays the index of the specified Help file. An application should use this value only for a Help file with a single index.
cdlHelpKey	&H101&	Displays Help for a particular keyword. When using this setting, you must also specify a keyword using the **HelpKey** property.
cdlHelpPartialKey	&H105&	Displays the topic found in the keyword list that matches the keyword passed in the dwData parameter if there is one exact match. If more than one match exists, the Search dialog box with the topics found listed in the Go To list box is displayed. If no match exists, the Search dialog box is displayed. To bring up the Search dialog box without passing a keyword, use a long pointer to an empty string.
cdlHelpQuit	&H2&	Notifies the Help application that the specified Help file is no longer in use.
cdlHelpSetContents	&H5&	Determines which contents topic is displayed when a user presses the F1 key.
cdlHelpSetIndex	&H5&	Sets the context specified by the **HelpContext** property as the current index for the Help file specified by the **HelpFile** property. This index remains current until the user accesses a different Help file. Use this value only for Help files with more than one index.

Remarks

The values for the **HelpCommand** property constants are listed in the Microsoft CommonDialog Control (MSComDlg) object library in the Object Browser.

The constant **cdlHelpContents** doesn't work for Help files created with Microsoft Help Workshop Version 4.0X. Instead, you use the value &HB to get the same effect. See the HelpCommand Property Example for a working code example.

Data Type

Integer

Visual Basic: CommonDialog Control

HelpCommand, Help Context Properties Example

The following example demonstrates several help commands. To try the example, place a **CommonDialog** control and five **CommandButton** controls on a form. Paste the code into the Declarations section, press F5 and click each button.

```
Option Explicit
Const HelpCNT = &HB

Private Sub Command1_Click()
   With CommonDialog1
      ' You must set the Help file name.
      ' Change this to an existing file on the hard disk.
      .HelpFile = "VB5.hlp"
      ' Display the Table of Contents. Note that the
      ' HelpCNT contstant is not an intrinsic
      ' constant. The cdlHelpSetContents ensures that
      ' only the Table of Contents (not Index or Find)
      ' shows.
      .HelpCommand = HelpCNT Or cdlHelpSetContents
      .ShowHelp
   End With

End Sub

Private Sub Command2_Click()
   With CommonDialog1
      .HelpFile = "VB5.hlp"
      ' Go to the Click Event topic in the Help file.
      ' The number is determined in the [MAP] section
      ' of the .HPJ file for the .chm file. You can
      ' edit this number only if you are using the
      ' Microsoft Help Workshop to build your
      ' own Help file.
      .HelpContext = 916302
      .HelpCommand = cdlHelpContext
      .ShowHelp
    End With
End Sub

Private Sub Command3_Click()
   With CommonDialog1
      .HelpFile = "VB5.hlp"
      ' Display help on Help.
      .HelpCommand = cdlHelpHelpOnHelp
      .ShowHelp
   End With
End Sub

Private Sub Command4_Click()
   With CommonDialog1
```

```
      .HelpFile = "VB5.hlp"
      .HelpKey = "data"
      ' Display the Index with the keyword selected.
      .HelpCommand = cdlHelpKey
      .ShowHelp
   End With
End Sub

Private Sub Command5_Click()
   With CommonDialog1
      .HelpFile = "VB5.hlp"
      .HelpKey = "arrays,"
      ' Display a list of topics found with
      ' the HelpKey.
      .HelpCommand = cdlHelpPartialKey
      .ShowHelp
   End With

End Sub

Private Sub Form_Load()
   ' Label the CommandButton controls.
   Command1.Caption = "Contents"
   Command2.Caption = "Specified Topic"
   Command3.Caption = "Help On Help"
   Command4.Caption = "Index of Topics"
   Command5.Caption = "Found Topics"
End Sub
```

Visual Basic: CommonDialog Control

Visual Studio 6.0

HelpContext Property (CommonDialog)

See Also Example Applies To

Returns or sets the context ID of the requested Help topic.

Syntax

object.**HelpContext** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string expression specifying the context ID of the requested Help topic.

Remarks

Use this property with the **HelpCommand** property (set **HelpCommand** = **cdlHelpContext**) to specify the Help topic to be displayed.

Data Type

Long

Visual Basic: CommonDialog Control

HelpCommand, Help Context Properties Example

The following example demonstrates several help commands. To try the example, place a **CommonDialog** control and five **CommandButton** controls on a form. Paste the code into the Declarations section, press F5 and click each button.

```
Option Explicit
Const HelpCNT = &HB

Private Sub Command1_Click()
   With CommonDialog1
      ' You must set the Help file name.
      ' Change this to an existing file on the hard disk.
      .HelpFile = "VB5.hlp"
      ' Display the Table of Contents. Note that the
      ' HelpCNT contstant is not an intrinsic
      ' constant. The cdlHelpSetContents ensures that
      ' only the Table of Contents (not Index or Find)
      ' shows.
      .HelpCommand = HelpCNT Or cdlHelpSetContents
      .ShowHelp
   End With

End Sub

Private Sub Command2_Click()
   With CommonDialog1
      .HelpFile = "VB5.hlp"
      ' Go to the Click Event topic in the Help file.
      ' The number is determined in the [MAP] section
      ' of the .HPJ file for the .chm file. You can
      ' edit this number only if you are using the
      ' Microsoft Help Workshop to build your
      ' own Help file.
      .HelpContext = 916302
      .HelpCommand = cdlHelpContext
      .ShowHelp
    End With
End Sub

Private Sub Command3_Click()
   With CommonDialog1
      .HelpFile = "VB5.hlp"
      ' Display help on Help.
      .HelpCommand = cdlHelpHelpOnHelp
      .ShowHelp
   End With
End Sub

Private Sub Command4_Click()
   With CommonDialog1
```

```
      .HelpFile = "VB5.hlp"
      .HelpKey = "data"
      ' Display the Index with the keyword selected.
      .HelpCommand = cdlHelpKey
      .ShowHelp
   End With
End Sub

Private Sub Command5_Click()
   With CommonDialog1
      .HelpFile = "VB5.hlp"
      .HelpKey = "arrays,"
      ' Display a list of topics found with
      ' the HelpKey.
      .HelpCommand = cdlHelpPartialKey
      .ShowHelp
   End With

End Sub

Private Sub Form_Load()
   ' Label the CommandButton controls.
   Command1.Caption = "Contents"
   Command2.Caption = "Specified Topic"
   Command3.Caption = "Help On Help"
   Command4.Caption = "Index of Topics"
   Command5.Caption = "Found Topics"
End Sub
```

Visual Basic for Applications Reference

Visual Studio 6.0

HelpContext Property

See Also Example Applies To Specifics

Returns or sets a string expression containing the context ID for a topic in a Help file. Read/write.

Remarks

The **HelpContext** property is used to automatically display the Help topic specified in the **HelpFile** property. If both **HelpFile** and **HelpContext** are empty, the value of **Number** is checked. If **Number** corresponds to a Visual Basic run-time error value, then the Visual Basic Help context ID for the error is used. If the **Number** value doesnt correspond to a Visual Basic error, the contents screen for the Visual Basic Help file is displayed.

Note You should write routines in your application to handle typical errors. When programming with an object, you can use the object's Help file to improve the quality of your error handling, or to display a meaningful message to your user if the error isnt recoverable.

Visual Basic for Applications Reference

HelpContext Property Example

This example uses the **HelpContext** property of the **Err** object to show the Visual Basic Help topic for the `Overflow` error.

```
Dim Msg
Err.Clear
On Error Resume Next
Err.Raise 6 ' Generate "Overflow" error.
If Err.Number <> 0 Then
   Msg = "Press F1 or HELP to see " & Err.HelpFile & " topic for" & _
   " the following HelpContext: " & Err.HelpContext
   MsgBox Msg, , "Error: " & Err.Description, Err.HelpFile, _
Err.HelpContext
End If
```

Visual Basic: RDO Data Control

Visual Studio 6.0

HelpContext, HelpFile Properties (Remote Data)

See Also Example Applies To

- **HelpContext** returns a context ID for a topic in a Microsoft Windows Help file.
- **HelpFile** returns a fully qualified path to the Help file as a variable.

Syntax

object.**HelpContext**

object.**HelpFile**

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

Return Values

The **HelpContext** property returns a **Long** value.

The **HelpFile** property returns a **String** value.

Remarks

If a Microsoft Windows Help file is specified in **HelpFile**, the **HelpContext** property is used to automatically display the Help topic it identifies.

Note You should write routines in your application to handle typical errors. When programming with an object, you can use the Help supplied by the object's Help file to improve the quality of your error handling, or to display a meaningful message to your user if the error is not recoverable.

Visual Basic Reference

Visual Studio 6.0

HelpContextID Property

See Also Example Applies To

Returns or sets an associated context number for an object. Used to provide context-sensitive Help for your application.

Syntax

object.**HelpContextID** [= *number*]

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

Part	**Description**
object	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*.
number	A numeric expression that specifies the context number of the Help topic associated with *object*.

Settings

The settings for *number* are:

Setting	**Description**
0	(Default) No context number specified.
> 0	An integer specifying a valid context number.

Remarks

For context-sensitive Help on an object in your application, you must assign the same context number to both *object* and to the associated Help topic when you compile your Help file.

If you've created a Microsoft Windows operating environment Help file for your application and set the application's **HelpFile** property, when a user presses the F1 key, Visual Basic automatically calls Help and searches for the topic identified by the current context number.

The current context number is the value of **HelpContextID** for the object that has the focus. If **HelpContextID** is set to 0, then Visual Basic looks in the **HelpContextID** of the object's container, and then that object's container, and so on. If a nonzero current context number can't be found, the F1 key is ignored.

For a **Menu** control, **HelpContextID** is normally read/write at run time. But **HelpContextID** is read-only for menu items that are exposed or supplied by Visual Basic to add-ins, such as the Add-In Manager command on the Add-Ins menu.

Visual Basic Reference

HelpContextID Property Example

This example uses topics in the Visual Basic Help file to demonstrate how to specify context numbers for Help topics. To try this example, paste the code into the Declarations section of a form that contains a **TextBox** control and a **Frame** control with an **OptionButton** control inside of it. Press F5. Once the program is running, move the focus to one of the controls, and press F1.

```
' Actual context numbers from the Visual Basic Help file.
Const winColorPalette = 21004   ' Define constants.
Const winToolbox = 21001
Const winCodeWindow = 21005

Private Sub Form_Load ()
   App.HelpFile = "VB.HLP"
   Frame1.HelpContextID = winColorPalette
   Text1.HelpContextID = winToolbox
   Form1.HelpContextID = winCodeWindow
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

HelpContextID Property (Toolbar Control)

See Also Example Applies To

Returns or sets an associated context number for an object. Used to provide context-sensitive Help for your application.

Note The **HelpContextID** property for the **Toolbar** control enables a link to Help from the Customize Toolbar dialog box rather than from the control itself. This behavior is different from that of other Visual Basic controls that contain the **HelpContextID** property.

Syntax

object.**HelpContextID** [= *number*]

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

Part	Description
object	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*.
number	A numeric expression that specifies the context number of the Help topic associated with *object*.

Settings

The settings for *number* are:

Setting	Description
0	(Default) No context number specified.
> 0	An integer specifying a valid context number.

Remarks

For context-sensitive Help on an object in your application, you must assign the same context number to both *object* and to the associated Help topic when you compile your Help file.

If you've created a Microsoft Windows operating environment Help file for your application and set the application's **HelpFile** property, when a user presses the F1 key, Visual Basic automatically calls Help and searches for the topic identified by the current context number.

The current context number is the value of **HelpContextID** for the object that has the focus. If **HelpContextID** is set to 0, then Visual Basic looks in the **HelpContextID** of the object's container, and then that object's container, and so on. If a nonzero current context number can't be found, the F1 key is ignored.

For a **Menu** control, **HelpContextID** is normally read/write at run time. But **HelpContextID** is read-only for menu items that are exposed or supplied by Visual Basic to add-ins, such as the Add-In Manager command on the Add-Ins menu.

Visual Basic Extensibility Reference

Visual Studio 6.0

HelpContextID Property (VBA Add-In Object Model)

See Also Example Applies To Specifics

Returns or sets a String containing the context ID for a topic in a Microsoft Windows Help file. Read/write.

Visual Basic Extensibility Reference

HelpContextID Property Example

The following example uses the **HelpContextID** property to return the context ID for the Help file corresponding to a project.

```
Debug.Print Application.VBE.VBProjects(1).HelpContextID
```

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

HelpFile Property

See Also Example Applies To Specifics

Returns or sets a string expression the fully qualified path to a Help file. Read/write.

Remarks

If a Help file is specified in **HelpFile**, it is automatically called when the user presses the **Help** button (or the F1 KEY) in the error message dialog box. If the **HelpContext** property contains a valid context ID for the specified file, that topic is automatically displayed. If no **HelpFile** is specified, the Visual Basic Help file is displayed.

Note You should write routines in your application to handle typical errors. When programming with an object, you can use the object's Help file to improve the quality of your error handling, or to display a meaningful message to your user if the error isnt recoverable.

Visual Basic for Applications Reference

HelpFile Property Example

This example uses the **HelpFile** property of the **Err** object to start the Help system. By default, the **HelpFile** property contains the name of the Visual Basic Help file.

```
Dim Msg
Err.Clear
On Error Resume Next    ' Suppress errors for demonstration purposes.
Err.Raise 6    ' Generate "Overflow" error.
Msg = "Press F1 or HELP to see " & Err.HelpFile & _
" topic for this error"
MsgBox Msg, , "Error: " & Err.Description,Err.HelpFile, Err.HelpContext
```

Visual Basic Reference

Visual Studio 6.0

HelpFile Property (App, CommonDialog, MenuLine)

See Also Example Applies To

Specifies the path and filename of a Help file used by your application to display Help or online documentation.

Syntax

object.**HelpFile**[= *filename*]

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

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
filename	A string expression specifying the path and filename of the help file for your application.

Remarks

If you've created a Help file for your application and set the application's **HelpFile** property, Visual Basic automatically calls Help when a user presses the F1 key. If there is a context number in the **HelpContextID** property for either the active control or the active form, Help displays a topic corresponding to the current Help context; otherwise it displays the main contents screen.

You can also use the **HelpFile** property to determine which Help file is displayed when a user requests Help from the Object Browser for an ActiveX component.

Visual Basic: Windows Controls

Visual Studio 6.0

HelpFile Property (Toolbar Control)

See Also Example Applies To

Specifies the path and filename of a Help file used by your application to display Help or online documentation.

Note The **HelpFile** property for the **Toolbar** control enables a link to Help from the Customize Toolbar dialog box rather than from the control itself. This behavior is different from that of other Visual Basic controls that contain the **HelpFile** property.

Syntax

object.**HelpFile**[= *filename*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
filename	A string expression specifying the path and filename of the Help file for your application.

Remarks

If you've created a Help file for your application and set the application's **HelpFile** property, Visual Basic automatically calls Help when a user presses the F1 key. If there is a context number in the **HelpContextID** property for either the active control or the active form, Help displays a topic corresponding to the current Help context; otherwise it displays the main contents screen.

You can also use the **HelpFile** property to determine which Help file is displayed when a user requests Help from the Object Browser for an ActiveX component.

Visual Basic Extensibility Reference

Visual Studio 6.0

HelpFile Property (VBA Add-In Object Model)

See Also Example Applies To Specifics

Returns or sets a String specifying the Microsoft Windows Help file for a project. Read/write.

Visual Basic Extensibility Reference

HelpFile Property Example

The following example uses the **HelpFile** property to assign a Help file to a project; the example verifies that the assignment was successful by printing the full path of the Help file.

```
Application.VBE.VBProjects(1).HelpFile = "C:\HelpStuff\veenob3.hlp"
Debug.Print Application.VBE.VBProjects(1).HelpFile
```

Visual Basic: CommonDialog Control

Visual Studio 6.0

HelpKey Property

See Also Example Applies To

Returns or sets the keyword that identifies the requested Help topic.

Syntax

object.**HelpKey** [= *string*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression specifying the keyword that identifies the Help topic.

Remarks

Use this property with the **HelpCommand** property (set **HelpCommand** = **cdlHelpKey**) to specify the Help topic to be displayed.

Data Type

String

Visual Basic: RDO Data Control

Visual Studio 6.0

hEnv Property (Remote Data)

See Also Example Applies To

Returns a value corresponding to the ODBC environment handle.

Syntax

object.**hEnv**

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

Return Values

The **hEnv** property returns a Long value containing the ODBC environment handle created by the ODBC driver manager corresponding to the specified **rdoEnvironment** object.

Remarks

This handle can be used to execute ODBC functions that require an ODBC **hEnv** environment handle.

Note While it is possible to execute ODBC API functions using the ODBC **hEnv, hDbc**, and **hStmt** handles, it is recommended that you do so with caution. Improper use of arbitrary ODBC API functions using these handles can result in unpredictable behavior. You should not attempt to save this handle in a variable for use at a later time as the value is subject to change.

Visual Basic: RDO Data Control

hEnv Property Example

The following example illustrates use of the **hEnv** property when accessing an ODBC API function. This code displays all registered data source names (DSNs) in a **ListBox** control.

```
Private Sub ShowDSNs_Click()
Dim fDirection As Integer
Dim szDSN As String * 1024
Dim cbDSNMax As Integer
Dim pcbDSN As Integer
Dim szDescription As String * 1024
Dim cbDescriptionMax As Integer
Dim pcbDescription As Integer
Dim Item As String
Set En = rdoEnvironments(0)
fDirection = SQL_FETCH_NEXT
cbDSNMax = 1023
cbDescriptionMax = 1023
List1.Clear
I = SQL_SUCCESS
 While I = SQL_SUCCESS
     szDSN = String(1024, " ")
     szDescription = String(1024, " ")
     I = SQLDataSources(En.hEnv, fDirection, szDSN, _
     cbDSNMax, pcbDSN, szDescription, _
     cbDescriptionMax, pcbDescription)
     Item = Left(szDSN, pcbDSN) & " - "  _
      & Left(szDescription, pcbDescription)
     Debug.Print Item
     List1.AddItem Item
 Wend

End Sub
```

Visual Basic Reference

Visual Studio 6.0

Hidden Property

See Also Example Applies To

Returns or sets the Hidden attribute of a **Member** object.

Syntax

object.**Hidden**

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

Hidden Property (MSChart)

See Also Example Applies To

Returns or sets a value that determines whether a series is displayed on the chart.

Syntax

object.**Hidden** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that controls whether a series is displayed on the chart, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The chart is drawn without displaying the series. However, any space allocated for the series still exists.
False	(Default) The series is displayed.

Visual Studio 6.0

Visual Basic: MSChart Control

Hide Property

See Also Example Applies To

Returns or sets a value that determines whether the axis on a chart is hidden.

Syntax

object.**Hide** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression that specifies whether the axis is hidden, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The axis scale, line, ticks and title are hidden.
False	(Default) The axis appears on the chart.

Visual Basic: Windows Controls

Visual Studio 6.0

HideColumnHeaders Property (ListView Control)

See Also Example Applies To

Returns or sets whether **ColumnHeader** objects in a **ListView** control are hidden in Report view.

Syntax

object.**HideColumnHeaders** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to a **ListView** control.
Boolean	A Boolean expression that specifies if the column headers are visible in Report view, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The column headers are not visible.
False	(Default) The column headers are visible.

Remarks

The **ListItem** objects and any related subitems remain visible even if the **HideColumnHeaders** property is set to **True**.

Visual Basic: Windows Controls

HideColumnHeaders Property Example

This example adds several **ListItem** objects with subitems to a **ListView** control. When you click on the **CommandButton**, the **HideColumnHeaders** property toggles between **True** (-1) and **False** (0). To try the example, place **ListView** and **CommandButton** controls on a form and paste the code into the form's Declarations section. Run the example and click the **CommandButton** to toggle the **HideColumnHeaders** property.

```
Private Sub Command1_Click()
   ' Toggle HideColumnHeaders property off and on.
   ListView1.HideColumnHeaders = Abs(ListView1.HideColumnHeaders) - 1
End Sub

Private Sub Form_Load()
   Dim clmX As ColumnHeader
   Dim itmX As ListItem
   Dim i As Integer
   Command1.Caption = "HideColumnHeaders"

   ' Add 3 ColumnHeader objects to the control.
   For i = 1 To 3
      Set clmX = ListView1.ColumnHeaders.Add()
      clmX.Text = "Col" & i
   Next I

   ' Set View to Report.
   ListView1.View = lvwReport

   ' Add 10 ListItems to the control.
   For i = 1 To 10
      Set itmX = ListView1.ListItems.Add()
      itmX.Text = "ListItem " & i
      itmX.SubItems(1) = "Subitem 1"
      itmX.SubItems(2) = "Subitem 2"
   Next i
End Sub
```

Visual Basic Reference

Visual Studio 6.0

HideSelection Property

See Also Example Applies To

Returns a value that determines whether selected text appears highlighted when a control loses the focus.

Syntax

object.**HideSelection**

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

Return Values

The **HideSelection** property return values are:

Value	Description
True	(Default) Selected text doesn't appear highlighted when the control loses the focus.
False	Selected text appears highlighted when the control loses the focus.

Remarks

You can use this property to indicate which text is highlighted while another form or a dialog box has the focus for example, in a spell-checking routine.

Visual Basic Reference

HideSelection Property Example

This example enables you to select text in each form and switch the focus between forms by clicking each form's title bar. The selection remains visible even when the form isn't active. To try the example, create two forms and draw a **TextBox** control on each. Set the **MultiLine** property to **True** for both **TextBox** controls, and set the **HideSelection** property to **False** for one of the **TextBox** controls. Paste the code into the Declarations section of both form modules, and then press F5.

```
Private Sub Form_Load ()
   Open "README.TXT" For Input As 1   ' Load file into text box.
   Text1.Text = Input$(LOF(1), 1)
   Close 1
   Form2.Visible = True   ' Load Form2, if not already loaded.
   ' Position forms side by side.
   Form1.Move 0, 1050, Screen.Width / 2, Screen.Height
   Form2.Move Screen.Width / 2, 1050, Screen.Width / 2, Screen.Height
   ' Enlarge text box to fill form.
   Text1.Move 0, 0, ScaleWidth, ScaleHeight
End Sub
```

Visual Basic Reference

Visual Studio 6.0

HideSelection Property (ActiveX Controls)

See Also Example Applies To

Returns a value that determines whether selected text appears highlighted when a control loses the focus.

Syntax

object.**HideSelection**

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

Return Values

The **HideSelection** property return values are:

Value	Description
True	(Default) Selected text doesn't appear highlighted when the control loses the focus.
False	Selected text appears highlighted when the control loses the focus.

Remarks

You can use this property to indicate which text is highlighted while another form or a dialog box has the focus for example, in a spell-checking routine.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

HighLight Property (MSHFlexGrid)

SeeAlso Example Applies To

Determines whether selected cells appear highlighted within the **MSHFlexGrid**.

Syntax

object.**HighLight** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer or constant that specifies when the **MSHFlexGrid** should highlight selected cells, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
flexHighlightNever	0	There is no highlight on the selected cells.
flexHighlightAlways	1	The selected cells are always highlighted. (Default)
flexHighlightWithFocus	2	The highlight appears only when the control has focus.

Remarks

When this property is set to zero and a range of cells is selected, there is no visual cue or emphasis indicating the selected cells.

Visual Basic: Windows Controls

Visual Studio 6.0

HighLighted Property

See Also Example Applies To

Returns or sets a value that determines if an object is highlighted.

Syntax

object.**HighLighted** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying if the object is highlighted, as shown in Settings.

Settings

The settings for *boolean* are:

Constant	Description
False	(Default). No highlighting occurs.
True	The item is highlighted.

Visual Basic: Windows Controls

Visual Studio 6.0

hImageList Property (ImageList Control)

See Also Example Applies To

Returns a handle to an **ImageList** control.

Syntax

object.**hImageList**

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

Remarks

The Microsoft Windows operating environment identifies an **ImageList** control in an application by assigning it a handle, or **hImageList**. The **hImageList** property is used with Windows API calls. Many ImageList-related API functions require the **hImageList** of the active window as an argument.

Note Because the value of this property can change while a program is running, never store the **hImageList** value in a variable.

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

hInstance Property

See Also Example Applies To

Returns a handle to the instance of the application.

Syntax

object.**hInstance**

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

Remarks

The **hInstance** property returns a Long data type.

When working with a project in the Visual Basic development environment, the **hInstance** property returns the instance handle of the Visual Basic instance.

Visual Basic: Internet Control

Visual Studio 6.0

hInternet Property

See Also Example Applies To

Returns the Internet handle from the underlying Wininet API. This handle can then be used in direct calls into the API. This property is not used when accessing the control from Visual Basic.

Syntax

*object.***hInternet**

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

Data Type

Long

Visual Basic Reference

Visual Studio 6.0

HitBehavior Property

See Also Example Applies To

Returns or sets a value that defines the hit testing behavior of the HitTest event on a Windowless **UserControl** object.

Syntax

object.**HitBehavior** [= *number*]

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

Part	Description
object	A UserControl object.
number	An integer that specifies hit testing behavior, as described in Settings.

Settings

The settings for *number* are:

Constant	Setting	Description
None	0	The HitTest event will always return a HitResult of 0 (vbHitResultOutside).
UseRegion	1	(Default) The HitTest event will return a HitResult of 3 (vbHitResultHit) when the cursor is over the MaskRegion of your control.
UsePaint	2	The HitTest event will return a HitResult of 3 (vbHitResultHit) when the cursor is over any painted area of the control.

Remarks

You can use the **HitBehavior** property to determine where hits will occur on your UserControl. By default, only the MaskRegion of the control will return a hit in the HitTest event. The MaskRegion consists of any subcontrols plus any mask defined by the MaskPicture and MaskColor properties. Any areas outside of the MaskRegion will return a HitResult of 0.

By setting **HitBehavior** to None, the HitTest event will never return a hit. When the **HitBehavior** is set to UsePaint, the mask region plus the areas painted by graphics methods will return a hit.

When used in combination with the **ClipBehavior** property, this property helps to determine the HitResult argument of the HitTest event.

Note This property is ignored if the **Windowless** property of the **UserControl** object is set to False or if the **BackStyle** property is set to Opaque.

Important Not all control containers support the Windowless property. The HitBehavior property should only be changed if you know it will be used in containers that support Windowless activation.

Visual Studio 6.0

Visual Basic: MSChart Control

HorzAlignment Property

See Also Example Applies To

Returns or sets the method of horizontal alignment of text.

Syntax

object.**HorzAlignment** [= *type*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
type	Integer. A VtHorizontalAlignment constant used to describe the horizontal alignment method of text.

Visual Basic Reference

Visual Studio 6.0

HostDataType Property

See Also Example Applies To

Returns or sets the data type to which the **DEParameter** object is converted.

Syntax

object.**HostDataType** [=*value*]

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

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
value	A constant or value that specifies the host data type. See Visual Basic's data types for information.

Visual Basic Reference

Visual Studio 6.0

HostName Property

See Also Example Applies To

Returns or sets the user-readable host name of your Visual Basic application.

Syntax

object.**HostName** [= *name*]

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

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
name	A string expression specifying the host name.

Remarks

When editing an object, the **HostName** property setting may be displayed in the object's window title. However, some applications that provide objects don't display **HostName**.

Visual Basic: Windows Controls

Visual Studio 6.0

HotImageList Property

See Also Example Applies To

Returns or sets the **ImageList** control to be used for "hot" imageswhich appear when the cursor hovers over a clickable spot and the **Style** property is set to **tbrTransparent**.

Syntax

object.**HotImageList** [= *imageList*]

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
imageList	An object reference that specifies which **ImageList** control to use for "hot" images.

Remarks

The **Button** object can display only one image for each button. At run time, it first determines how the button should be drawn (i.e., normal, 'hot' or disabled) and then uses the image from the appropriate image list (**ImageList**, **DisabledImageList** or **HotImageList**) using the sole **Image** property as the key. It is important to understand that related images in each of the three image lists must be consistently named so that the **Toolbar** control pulls the correct ones. For example, if a particular button is making use of all three image types, then each of the three images must be defined in their respective image lists to have either the same **Index** as the other two or the same **Key**.

Visual Basic: Windows Controls

DisabledImageList, HotImageList, ImageList Properties Example

The following example demonstrates the use of the **DisableImageList**, **HotImageList**, and **ImageList** properties. Notice that all three image lists use the same **Key** property. The example assumes that three **ImageList** controls named imgNormal, imgHot, and imgDisabled have been placed on a form.

```
Private Sub Form_Load()
   Dim str As String
   str = "D:\VB\Icons\Misc\" ' Change to match your graphics location.
   imgNormal.ListImages.Add Key:="face", _
   Picture:=LoadPicture(str & "Face02.ico")
   imgHot.ListImages.Add Key:="face", _
   Picture:=LoadPicture(str & "Face03.ico")
   imgDisabled.ListImages.Add Key:="face", _
   Picture:=LoadPicture(str & "Face01.ico")

   ' Set the ImageList, DisableImageList, and HotImageList properties.
   With Toolbar1
      .ImageList = imgNormal
      .DisabledImageList = imgDisabled
      .HotImageList = imgHot
      .Buttons.Add Caption:="Sad", Image:="face", Style:=tbrCheck
      .Buttons.Add Caption:="Happy", Image:="face", Style:=tbrCheck
      .Buttons.Add Caption:="Face", Image:="face", Style:=tbrCheck

      .Buttons(1).Enabled = False
      .Buttons(3).Value = tbrPressed
   End With
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

HotTracking Property

See Also Example Applies To

Returns a value that determines whether mouse-sensitive highlighting is enabled.

Syntax

object.**HotTracking**

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

Return Values

The return settings for are:

Setting	Description
True	Hot tracking is enabled. Header captions are highlighted as the mouse pointer passes over them.
False	(Default) The **HeaderItem** objects do not respond to mouse movement unless a mouse button is clicked.

Remarks

Hot tracking is a feature that provides feedback to the user when the mouse pointer passes over the control. With **HotTracking** set to **True**, the control responds to mouse movement by highlighting the header over which the mouse pointer is positioned.

Visual Basic: Windows Controls

Visual Studio 6.0

Hour Property

See Also Example Applies To

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

Syntax

object.**Hour** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A numeric expression that specifies the current hour that is displayed.

Remarks

The **Hour** property can be any integer from 0 to 23.

Visual Basic: Windows Controls

Visual Studio 6.0

HoverSelection Property

See Also Example Applies To

Returns or sets a value that determines if a **ListItem** object is selected when the mouse pointer hovers over it.

Syntax

object.**HoverSelection** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
boolean	A Boolean expression specifying if the object is selected, as shown in Settings.

Settings

The settings for *boolean* are:

Constant	Description
False	(Default). No selection occurs.
True	The item is selected after the mouse pointer hovers over it for a few seconds.

Visual Basic Reference

Visual Studio 6.0

hPal Property

See Also Example Applies To

Returns or sets a handle to the palette of a picture in a **Picture** object.

Syntax

object.**hPal** [= *value*]

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

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
value	The handle to the palette for the picture (HPAL).

Remarks

The **hPal** property is useful when you need to pass a handle to a palette as part of a call to a function in a dynamic-link library (DLL) or the Windows API.

Visual Basic Reference

Visual Studio 6.0

HScrollSmallChange, VScrollSmallChange Properties

See Also Example Applies To

Returns or sets the distance the **UserDocument** will scroll when the user clicks a scroll arrow.

Syntax

object.**HScrollSmallChange** = *single*

object.**VScrollSmallChange** = *single*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
single	The distance in twips the **UserDocument** will scroll when the user clicks the scroll arrow.

Remarks

There is no LargeChange property counterpart to the **HScrollSmallChange** and **VScrollSmallChange** properties. The LargeChange is determined by the **ViewPort** objects **ViewPortHeight** and **ViewPortWidth** properties.

Visual Basic: RDO Data Control

Visual Studio 6.0

hStmt Property (Remote Data)

See Also Example Applies To

Returns a value corresponding to the ODBC statement handle.

Syntax

object.**hStmt**

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

Return Values

The **hStmt** property returns a Long value containing the ODBC statement handle created by the ODBC driver manager corresponding to the specified **rdoResultset** object.

Remarks

This handle can be used to execute ODBC functions that require an ODBC **hStmt** statement handle.

Note While it is possible to execute ODBC API functions using the ODBC **hEnv, hDbc**, and **hStmt** handles, it is recommended that you do so with caution. Improper use of arbitrary ODBC API functions using these handles can result in unpredictable behavior. You should not attempt to save this handle in a variable for use at a later time as the value is subject to change.

Visual Basic: RDO Data Control

hStmt Property Example

This example illustrates use of the **hStmt** property to return a configuration option for a specific statement handle. The example uses the **SQLGetStmtOption** function to determine the type of cursor created by the **OpenResultset** method. Note that this value is also supplied by the **rdoResultset Type** property.

```
Option Explicit
Dim en As rdoEnvironment
Dim cn As rdoConnection
Dim rs As rdoResultset
Dim rc As Integer
Dim CursorType As Long
Dim T As String

'Declare Function SQLGetStmtOption Lib "odbc32.dll" (ByVal hstmt&, ByVal fOption%, ByRef pvParam As Any) As Integer

Private Sub Form_Load()
Set en = rdoEngine.rdoEnvironments(0)

en.CursorDriver = rdUseOdbc

Set cn = en.OpenConnection(dsName:="WorkDB", _
    prompt:=rdDriverNoPrompt, _
    Connect:="Uid=;pwd=;database=Pubs")

Set rs = cn.OpenResultset("Select * from Publishers", _
 rdOpenKeyset, rdConcurRowVer)

Select Case rs.Type
    Case rdOpenForwardOnly: T = "Forward-only"
    Case rdOpenStatic: T = "Static"
    Case rdOpenKeyset: T = "Keyset"
    Case rdOpenDynamic: T = "Dynamic"
End Select
MsgBox "RDO indicates that a " & T  _
   & " Cursor was created"
CursorType = 0
rc = SQLGetStmtOption(rs.hStmt,  _
   SQL_CURSOR_TYPE, CursorType)

Select Case CursorType
    Case SQL_CURSOR_FORWARD_ONLY: T = "Forward-only"
    Case SQL_CURSOR_STATIC: T = "Static"
    Case SQL_CURSOR_KEYSET_DRIVEN: T = "Keyset"
    Case SQL_CURSOR_DYNAMIC: T = "Dynamic"
End Select
MsgBox "ODBC indicates that a " & T  _
   & " Cursor was created"
End Sub
```

Visual Basic Reference

Visual Studio 6.0

hWnd Property

See Also Example Applies To

Returns a handle to a form or control.

Note This property is not supported for the **OLE** container control.

Syntax

object.**hWnd**

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

Remarks

The Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or **hWnd**. The **hWnd** property is used with Windows API calls. Many Windows operating environment functions require the **hWnd** of the active window as an argument.

Note Because the value of this property can change while a program is running, never store the **hWnd** value in a variable.

If the **Windowless** property of a **User Control** is set to **True**, the **hWnd** property will return 0.

Visual Basic Reference

hWnd Property Example

This example forces a form to always remain on top. To try this example, create a form (not an MDI child form), and then create a menu for the form called Main. Insert a submenu in it called Always On Top, and set its Name to mnuTopmost. Create a new module using the Add Module command on the Project menu. Paste the **Declare** statement into the Declarations section of the new module, being sure that the statement is on one line with no break or wordwrap. Then paste the **Sub** procedure into the Declarations section of the form and press F5.

```
' Declaration of a Windows routine.
' This statement should be placed in the module.
Declare Function SetWindowPos Lib "user32" Alias _
  "SetWindowPos" (ByVal hwnd As Long, ByVal _
  hWndInsertAfter As Long, ByVal x As Long, ByVal y As _
  Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags _
  As Long) As Long

' Set some constant values (from WIN32API.TXT).
Const conHwndTopmost = -1
Const conHwndNoTopmost = -2
Const conSwpNoActivate = &H10
Const conSwpShowWindow = &H40

Private Sub mnuTopmost_Click ()
   ' Add or remove the check mark from the menu.
   mnuTopmost.Checked = Not mnuTopmost.Checked
   If mnuTopmost.Checked Then
      ' Turn on the TopMost attribute.
      SetWindowPos hWnd, conHwndTopmost, 0, 0, 0, 0, _
        conSwpNoActivate Or conSwpShowWindow
   Else
      ' Turn off the TopMost attribute.
      SetWindowPos hWnd, conHwndNoTopmost, 0, 0, 0, _
        0, conSwpNoActivate Or conSwpShowWindow
   End If
End Sub
```

This example automatically drops down the list portion of a **ComboBox** control whenever the **ComboBox** receives the focus. To try this example, create a new form containing a **ComboBox** control and an **OptionButton** control (used only to receive the focus). Create a new module using the Add Module command on the Project menu. Paste the **Declare** statement into the Declarations section of the new module, being sure that the statement is on one line with no break or wordwrap. Then paste the **Sub** procedure into the Declarations section of the form, and press F5. Use the TAB key to move the focus to and from the **ComboBox**.

```
Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
  (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As _
  Long, lParam As Long) As Long

Private Sub Combo1_GotFocus ()
   Const CB_SHOWDROPDOWN = &H14F
   Dim Tmp
   Tmp = SendMessage(Combo1.hWnd, CB_SHOWDROPDOWN, 1, ByVal 0&)
End Sub
```

Visual Basic Reference

Visual Studio 6.0

hWnd Property (ActiveX Controls)

See Also Example Applies To

Returns a handle to a form or control.

Note This property is not supported for the **OLE** container control.

Syntax

object.**hWnd**

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

Remarks

The Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or **hWnd**. The **hWnd** property is used with Windows API calls. Many Windows operating environment functions require the **hWnd** of the active window as an argument.

Note Because the value of this property can change while a program is running, never store the **hWnd** value in a variable.

Visual Basic: Multimedia MCI Control

Visual Studio 6.0

hWndDisplay Property (Multimedia MCI Control)

See Also Example Applies To

Specifies the output window for MCI MMMovie or Overlay devices that use a window to display output. This property is not available at design time.

Syntax

[*form*.]*MMControl*.**hWndDisplay**

Remarks

This property is a handle to the window that the MCI device uses for output. If the handle is 0, a default window (also known as the "stage window") is used.

To determine whether a device uses this property, look at the **UsesWindows** property settings.

In Visual Basic, to get a handle to a control, first use the **SetFocus** method to set the focus to the desired control. Then call the Windows **GetFocus** function.

To get a handle to a Visual Basic form, use the **hWnd** property for that form.

Data Type

Integer

Visual Basic: DataGrid Control

Visual Studio 6.0

hWndEditor Property

See Also Example Applies To

Returns the unique window handle assigned to a **DataGrid** control's editing window by the Microsoft Windows operating environment. Not available at design time.

Syntax

object.**hWndEditor**

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

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

Remarks

Experienced users can pass the value of this property to Windows API calls that require a valid window handle.

When editing is not in progress, this property returns 0.

Note Since the value of this property can change while a program is running, never store the **hWndEditor** value in a variable. Also, do not use the **hWndEditor** property to test whether editing is in progress. The **EditActive** property is provided for this purpose.

Visual Basic Reference

Visual Studio 6.0

Hyperlink Property

See Also Example Applies To

Returns a reference to the **Hyperlink** object.

Syntax

object.**Hyperlink**

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