Visual Basic Reference

Visual Studio 6.0

AccessKeys Property

See Also Example Applies To

Returns or sets a string that contains the keys that will act as the access keys (or hot keys) for the control.

Syntax

object.**AccessKeys** [= *AccessKeyString*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
AccessKeyString	A string containing the keys that will act as the access keys.

Remarks

The **AccessKeys** property is a string that contains all the access keys for the control. As an example, to set the letters S and Y as the access keys, the **AccessKeys** property would be set to "sy".

When a user presses one of the access keys in conjunction with the ALT key, the control will get the focus (depending on the setting of the **ForwardFocus** property).

Access keys for constituent controls are implicitly included as AccessKeys, although they will not appear in the **AccessKeys** property.

Visual Basic: Internet Control

Visual Studio 6.0

AccessType Property

See Also Example Applies To

Sets or returns a value that determines the type of access (through a proxy or directly) that the control will use to communicate with the Internet. This value can be changed while an asynchronous request is being processed, but will not take effect until the next connection is established.

Syntax

object.**AccessType** = *type*

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
type	Integer (enumerated). A numeric expression that determines the type of access used, as described in Settings.

Settings

Valid settings for *type* are:

Constant	Value	Description
icUseDefault	0	Default. Use Defaults. The control uses default settings found in the registry to access the Internet.
icDirect	1	Direct to Internet. The control has a direct connection to the Internet.
icNamedProxy	2	Named Proxy. Instructs the control to use the proxy server specified in the **Proxy** property.

Visual Basic: SysInfo Control

Visual Studio 6.0

ACStatus Property

See Also Example Applies To

Returns a value that indicates whether or not the system is using AC power.

Syntax

object.**ACStatus**

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

Settings

The **ACStatus** property settings are:

Setting	Description
0	The system is not using AC power.
1	The system is using AC power.
255	AC power status is unknown.

Visual Basic: SysInfo Control

ACStatus Property Example

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

```
Private Sub Timer1_Timer()
   Select Case SysInfo1.ACStatus
      Case 0
         Label1.Caption = "AC Power: Off"
      Case 1
         Label1.Caption = "AC Power: On"
      Case 255
         Label1.Caption = "AC Power: Unknown"
   End Select
End Sub
```

Visual Basic: CommonDialog Control

Visual Studio 6.0

Action Property (CommonDialog)

See Also Example Applies To

Returns or sets the type of dialog box to be displayed. Not available at design time.

Note The **Action** property is included for compatibility with earlier versions of Visual Basic. For additional functionality, use the following new methods: **ShowColor**, **ShowFont**, **ShowHelp**, **ShowOpen**, **ShowPrinter**, and **ShowSave**.

Syntax

*object***.Action** [= *value*]

The **Action** 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 specifying the type of dialog box displayed, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
0	No Action.
1	Displays Open dialog box.
2	Displays Save As dialog box.
3	Displays Color dialog box.
4	Displays Font dialog box.
5	Displays Printer dialog box.
6	Runs WINHLP32.EXE.

Data Type

Integer

Visual Basic: MAPI Controls

Visual Studio 6.0

Action Property (MAPIMessages Control)

See Also Example Applies To

Determines what action is performed when the **MAPIMessages** control is invoked. This property is not available at design time. Setting the **Action** property at run time invokes the control. This property is write-only at run time.

Note The **Action** property is included for compatibility with earlier versions of Visual Basic. For additional functionality, use the new methods listed in the Methods table for the **MAPIMessages** control.

Syntax

```
object.Action [ = value ]
```

The **Action** 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 action to perform.

Remarks

The following table lists the **Action** property settings used for backwards compatibility and the corresponding new methods.

Action property setting	Corresponding method
MESSAGE_FETCH	Fetch method
MESSAGE_SENDDLG	Send method
MESSAGE_SEND	Send method
MESSAGE_SAVEMSG	Save method
MESSAGE_COPY	Copy method
MESSAGE_COMPOSE	Compose method
MESSAGE_REPLY	Reply method

MESSAGE_REPLYALL	ReplyAll method
MESSAGE_FORWARD	Forward method
MESSAGE_DELETE	Delete method
MESSAGE_SHOWADBOOK	Show method
MESSAGE_SHOWDETAILS	Show method
MESSAGE_RESOLVENAME	ResolveName method
RECIPIENT_DELETE	Delete method
ATTACHMENT_DELETE	Delete method

Data Type

Integer

Visual Basic: MAPI Controls

Visual Studio 6.0

Action Property (MAPISession Control)

See Also Example Applies To

Determines what action is performed when the **MAPISession** control is invoked. This property is not available at design time. Setting the **Action** property at run time invokes the control. The **Action** property is write-only at run time.

Note The **Action** property is included for compatibility with earlier versions of Visual Basic. For additional functionality, use the new methods listed in the Methods list for the **MAPISession** control.

Syntax

```
object.Action [ = value ]
```

The **Action** 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 action to perform, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
mapSignOn	1	Logs user into the account specified by the **UserName** and **Password** properties and provides a session handle to the underlying message subsystem. The session handle is stored in the **SessionID** property. Depending on the value of the **NewSession** property, the session handle may refer to a newly created session or an existing session.
mapSignOff	2	Ends the messaging session and signs the user off the specified account.

Remarks

This property is used to select between signing on and signing off from a messaging session. When signing on, a session handle is returned in the **SessionID** property.

Data Type

Integer

Visual Basic Reference

Visual Studio 6.0

Action Property (OLE Container)

See Also Example Applies To

Sets a value that determines an action. Not available at design time.

Note The **Action** property is included for compatibility with earlier versions. For current functionality, use the methods listed in Settings.

Syntax

object.**Action** = *value*

The **Action** 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 integer specifying the type of action, as described in Settings.

Settings

The settings for *value* are:

Value	Description	Current method
0	Creates an embedded object.	**CreateEmbed**
1	Creates a linked object from the contents of a file.	**CreateLink**
4	Copies the object to the system Clipboard.	**Copy**
5	Copies data from the system Clipboard to an **OLE** container control.	**Paste**
6	Retrieves the current data from the application that supplied the object and displays that data as a picture in the **OLE** container control.	**Update**
7	Opens an object for an operation, such as editing.	**DoVerb**
9	Closes an object and terminates the connection to the application that provided the object.	**Close**

10	Deletes the specified object and frees the memory associated with it.	**Delete**
11	Saves an object to a data file.	**SaveToFile**
12	Loads an object that was saved to a data file.	**ReadFromFile**
14	Displays the Insert Object dialog box.	**InsertObjDlg**
15	Displays the Paste Special dialog box.	**PasteSpecialDlg**
17	Updates the list of verbs an object supports.	**FetchVerbs**

18	Saves an object to the OLE version 1.0 file format.	**SaveToOle1File**

Visual Basic Extensibility Reference

Visual Studio 6.0

ActiveCodePane Property

See Also Example Applies To Specifics

Returns the active or last active **CodePane** object or sets the active **CodePane** object. Read/write.

Remarks

You can set the **ActiveCodePane** property to any valid **CodePane** object, as shown in the following example:

```
Set MyApp.VBE.ActiveCodePane = MyApp.VBE.CodePanes(1)
```

The preceding example sets the first code pane in a collection of code panes to be the active code pane. You can also activate a code pane using the **Set** method.

Visual Basic Extensibility Reference

ActiveCodePane Property Example

The following example uses the **ActiveCodePane** property and **TopLine** properties to obtain the number of the top line in the active code pane.

```
Debug.Print Application.VBE.ActiveCodePane.TopLine
```

Visual Basic: RDO Data Control

Visual Studio 6.0

ActiveConnection Property

See Also Example Applies To

Returns or sets an object reference indicating the connection this query should be associated with.

Syntax

object.**ActiveConnection** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An expression that evaluates to a valid **rdoConnection** or derived object. *Value* defaults to the **rdoConnection** used to create the object or **Nothing**.

Remarks

The **ActiveConnection** property holds a reference to the connection associated with the **rdoQuery** or **rdoResultset** object. All database statements executed by the object are executed against this connection.

When working with an **rdoQuery** object, the **ActiveConnection** property can be set to **Nothing** which dissociates the object from a specific connection. You can subsequently re-associate the **rdoQuery** object to another **rdoConnection** object by setting the **ActiveConnection** object. Using this technique, a query can be executed against a set of connections.

When working with the **rdoResultset** object and the Client Batch cursor library, you can set the **ActiveConnection** property to **Nothing**. In this case, if the result set is created with a static cursor and the **rdConcurBatch** concurrency option, the **rdoResultset** data is still available and you are free to make changes or additions to the result set. Once you set the **ActiveConnection** to an open **rdoConnection** object, you can use the **BatchUpdate** method to post these changes to the remote database.

Visual Basic: RDO Data Control

ActiveConnection Property Example

The following examples illustrates use of the **ActiveConnection** property to select an **rdoConnection**. In this case, the application opens two separate connections and uses the same **rdoQuery** against each.

```
Dim rdoCn As New rdoConnection
Dim rdoCn2 As New rdoConnection
Dim rdoQy As New rdoQuery
Dim rdoRs As rdoResultset
Dim rdoCol As rdoColumn
Dim rdoEn As rdoEnvironment

Private Sub Form_Load()
On Error GoTo CnEh

Set rdoEn = rdoEnvironments(0)

With rdoCn

    .Connect = "UID=;PWD=;Database=WorkDB;" _
        & "Server=Betav486;Driver={SQL Server}" _
        & "DSN='';"
    .LoginTimeout = 5
    .EstablishConnection rdDriverNoPrompt, True
    rdoEn.rdoConnections.Add rdoCn
End With

With rdoCn2
    .Connect = "UID=;PWD=;Database=Pubs;" _
        & "Server=Betav486;Driver={SQL Server}" _
        & "DSN='';"
    .LoginTimeout = 5
    .EstablishConnection rdDriverNoPrompt, True
    rdoEn.rdoConnections.Add rdoCn2
End With

With rdoQy
    Set .ActiveConnection = rdoCn
    .SQL = "Select Name, refDate " _
      & " from Sysobjects where type = 'U' "
    .LockType = rdConcurReadOnly
    .RowsetSize = 1
    .CursorType = rdUseServer
End With

For Each rdoCn In rdoEn.rdoConnections
    Set rdoQy.ActiveConnection = rdoCn
    Set rdoRs = rdoQy.OpenResultset(rdOpenForwardOnly)
    With rdoRs
        For Each rdoCol In rdoRs.rdoColumns
            Debug.Print rdoCol.Name,
        Next
        Debug.Print
        Do Until rdoRs.EOF
                For Each rdoCol In rdoRs.rdoColumns
                    Debug.Print rdoCol
                Next
            rdoRs.MoveNext
        Loop
    End With
Next                    ' Next Connection

Exit Sub
CnEh:
Dim er As rdoError
    Debug.Print Err, Error
    For Each er In rdoErrors
        Debug.Print er.Description, er.Number
    Next er
    Resume Next
End Sub
```

Visual Basic Reference

Visual Studio 6.0

ActiveControl Property

See Also Example Applies To

Returns the control that has the focus. When a form is referenced, as in `ChildForm.ActiveControl`, **ActiveControl** specifies the control that would have the focus if the referenced form were active. Not available at design time; read-only at run time.

Syntax

object.**ActiveControl**

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

Remarks

You can use **ActiveControl** to access a control's properties or to invoke its methods: For example, `Screen.ActiveControl.Tag = "0"`. A run-time error occurs if all controls on the form are invisible or disabled.

Each form can have an active control (`Form.ActiveControl`), regardless of whether or not the form is active. You can write code that manipulates the active control on each form in your application even when the form isn't the active form.

This property is especially useful in a multiple-document interface (MDI) application where a button on a toolbar must initiate an action on a control in an MDI child form. When a user clicks the Copy button on the toolbar, your code can reference the text in the active control on the MDI child form, as in `ActiveForm.ActiveControl.SelText`.

Note If you plan to pass `Screen.ActiveControl` to a procedure, you must declare the argument in that procedure with the clause `As Control` rather than specifying a control type (`As TextBox` or `As ListBox`) even if `ActiveControl` always refers to the same type of control.

Visual Basic Reference

ActiveControl Property Example

This example displays the text of the active control. To try this example, paste the code into the Declarations section of a form that contains **TextBox**, **Label**, and **CommandButton** controls, and then press F5 and click the form.

```
Private Sub Form_Click ()
   If TypeOf Screen.ActiveControl Is TextBox Then
      Label1.Caption = Screen.ActiveControl.Text
   Else
      Label1.Caption = "Button: " + Screen.ActiveControl.Caption
   End If
End Sub
```

This example shows how you can use the **Clipboard** object in cut, copy, paste, and delete operations using buttons on a toolbar. To try this example, put **TextBox** and **CheckBox** controls on Form1, and then create a new MDI form. On the MDI form, insert a **PictureBox** control, and then insert a **CommandButton** in the **PictureBox**. Set the **Index** property of the **CommandButton** to 0 (creating a control array). Set the **MDIChild** property of Form1 to **True**.

To run the example, copy the code into the Declarations section of the **MDIForm**, and then press F5. Notice that when the **CheckBox** has the focus, the buttons don't work, since the **CheckBox** is now the active control instead of the **TextBox**.

```
Private Sub MDIForm_Load ()
   Dim I   ' Declare variable.
   Command1(0).Move 0, 0, 700, 300   ' Position button on toolbar.
   For I = 1 To 3   ' Create other buttons.
      Load Command1(I)   ' Create button.
      Command1(I).Move I * 700, 0, 700, 300  ' Place and size button.
      Command1(I).Visible = True   ' Display button.
   Next I
   Command1(0).Caption = "Cut"   ' Set button captions.
   Command1(1).Caption = "Copy"
   Command1(2).Caption = "Paste"
   Command1(3).Caption = "Del"
End Sub

Private Sub Command1_Click (Index As Integer)
   ' ActiveForm refers to the active form in the MDI form.
   If TypeOf ActiveForm.ActiveControl Is TextBox Then
      Select Case Index
         Case 0   ' Cut.
            ' Copy selected text onto Clipboard.
            Clipboard.SetText ActiveForm.ActiveControl.SelText
            ' Delete selected text.
            ActiveForm.ActiveControl.SelText = ""
         Case 1   ' Copy.
            ' Copy selected text onto Clipboard.
            Clipboard.SetText ActiveForm.ActiveControl.SelText
         Case 2   ' Paste.
            ' Put Clipboard text in text box.
            ActiveForm.ActiveControl.SelText = Clipboard.GetText()
         Case 3   ' Delete.
            ' Delete selected text.
            ActiveForm.ActiveControl.SelText = ""
      End Select
   End If
End Sub
```

Visual Basic Reference

Visual Studio 6.0

ActiveForm Property

See Also Example Applies To

Returns the form that is the active window. If an **MDIForm** object is active or is referenced, it specifies the active MDI child form.

Syntax

object.**ActiveForm**

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

Remarks

Use the **ActiveForm** property to access a form's properties or to invoke its methods for example, `Screen.ActiveForm.MousePointer = 4`.

This property is especially useful in a multiple-document interface (MDI) application where a button on a toolbar must initiate an action on a control in an MDI child form. When a user clicks the Copy button on the toolbar, your code can reference the text in the active control on the MDI child form for example, `ActiveForm.ActiveControl.SelText`.

When a control on a form has the focus, that form is the active form on the screen (`Screen.ActiveForm`). In addition, an **MDIForm** object can contain one child form that is the active form within the context of the MDI parent form (`MDIForm.ActiveForm`). The **ActiveForm** on the screen isn't necessarily the same as the **ActiveForm** in the MDI form, such as when a dialog box is active. For this reason, specify the **MDIForm** with **ActiveForm** when there is a chance of a dialog box being the **ActiveForm** property setting.

Note When an active MDI child form isn't maximized, the title bars of both the parent form and the child form appear active.

If you plan to pass `Screen.ActiveForm` or `MDIForm.ActiveForm` to a procedure, you must declare the argument in that procedure with the generic type (`As Form`) rather than a specific form type (`As MyForm`) even if **ActiveForm** always refers to the same type of form.

The **ActiveForm** property determines the default value for the **ProjectTemplate** object.

Visual Basic Reference

ActiveForm Property Example

This example prints the time on the active child form in an **MDIForm** object. To try this example, create an **MDIForm**, draw a **PictureBox** control on it and a **CommandButton** control in the **PictureBox**. In Form1, set the **MDIChild** property to **True**. (You can also set **AutoRedraw** to **True** to keep text on the form even after covering it with another form.) Paste the appropriate code into the Declarations section of each form, and then press F5.

```
' Copy all code into the MDI form.
Private Sub MDIForm_Load ()
   Dim NewForm As New Form1   ' Create new instance of Form1.
   NewForm.Show
End Sub

Private Sub Command1_Click ()
   ' Print the time on the active form.
   ActiveForm.Print "The time is " & Format(Now, "Long Time")
End Sub
```

This example shows how you can use the **Clipboard** object in cut, copy, paste, and delete operations using buttons on a toolbar. To try this example, create a new project, then put **TextBox** and **CheckBox** controls on Form1, and then create a new MDI form. On the MDI form, place a **PictureBox** control, and then insert a **CommandButton** control in the **PictureBox**. Set the **Index** property of the **CommandButton** to 0 (creating a control array). Set the **MDIChild** property of Form1 to **True**.

To run the example, copy the code into the Declarations section of the **MDIForm**, and then press F5. Notice that when the **CheckBox** has the focus, the buttons don't work, since the **CheckBox** is now the active control instead of the **TextBox**.

```
Private Sub MDIForm_Load ()
   Dim I         ' Declare variable.
   Command1(0).Move 0, 0, 700, 300   ' Position button on toolbar.
   For I = 1 To 3   ' Create other buttons.
      Load Command1(I)   ' Create button.
      Command1(I).Move I * 700, 0, 700, 300 ' Place and size button.
      Command1(I).Visible = True   ' Display button.
   Next I
   Command1(0).Caption = "Cut"   ' Set button captions.
   Command1(1).Caption = "Copy"
   Command1(2).Caption = "Paste"
   Command1(3).Caption = "Del"
End Sub

Private Sub Command1_Click (Index As Integer)
   ' ActiveForm refers to the active form in the MDI form.
   If TypeOf ActiveForm.ActiveControl Is TextBox Then
      Select Case Index
         Case 0   ' Cut.
            ' Copy selected text to Clipboard.
            Clipboard.SetText ActiveForm.ActiveControl.SelText
            ' Delete selected text.
            ActiveForm.ActiveControl.SelText = ""
         Case 1   ' Copy.
            ' Copy selected text to Clipboard.
            Clipboard.SetText ActiveForm.ActiveControl.SelText
         Case 2   ' Paste.
            ' Put Clipboard text in text box.
            ActiveForm.ActiveControl.SelText = Clipboard.GetText()
         Case 3   ' Delete.
            ' Delete selected text.
            ActiveForm.ActiveControl.SelText = ""
      End Select
   End If
End Sub
```

Visual Basic: DataRepeater Control

Visual Studio 6.0

ActiveRow Property

See Also Example Applies To

Returns or sets the row index where the current record will be positioned. The ActiveRowChanged event occurs when setting this property.

Syntax

object.**ActiveRow** [=*integer*]

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
integer	A numeric expression that specifies the row where the current record is placed.

Remarks

The property value must be between 1 and the value of the **VisibleRows** property.

The row index is 1-based, and setting the property to 0 is not allowed. However, when 0 is returned, it indicates that the current record is scrolled out of view.

When the **DataRepeater** control displays the beginning or end of a recordset, it's possible that the **ActiveRow** won't have a logical setting. For example, if the current record is the last in the recordset, setting the **ActiveRow** to 2 or 1 when the **VisibleRows** is 3, is an invalid setting.

Visual Studio 6.0

Visual Basic: MSChart Control

ActiveSeriesCount Property

See Also Example Applies To

Returns the number of series that appear on a chart based on the number of columns in the **DataGrid** object and the type of chart being drawn.

Syntax

*object.***ActiveSeriesCount**

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

Visual Basic Extensibility Reference

Visual Studio 6.0

ActiveVBProject Property

See Also Example Applies To Specifics

Returns the active project in the Project window. Read-only.

Remarks

The **ActiveVBProject** property returns the project that is selected in the **Project** window or the project in which the components are selected. In the latter case, the project itself isn't necessarily selected. Whether or not the project is explicitly selected, there is always an active project.

Visual Basic Extensibility Reference

ActiveVBProject Property Example

The following example uses the **ActiveVBProject** property to return the name of the active project.

```
Debug.Print Application.VBE.ActiveVBProject.Name
```

Visual Basic Extensibility Reference

Visual Studio 6.0

ActiveWindow Property

See Also Example Applies To Specifics

Returns the active window in the development environment. Read-only.

Remarks

When more than one window is open in the development environment, the **ActiveWindow** property setting is the window with the focus. If the main window has the focus, **ActiveWindow** returns **Nothing**.

Visual Basic Extensibility Reference

ActiveWindow Property Example

The following example uses the **ActiveWindow** property to return the caption of the active window.

```
Debug.Print Application.VBE.ActiveWindow.Caption
```

Visual Basic Extensibility Reference

Visual Studio 6.0

AddIns Property

See Also Example Applies To Specifics

Returns a collection which add-ins can use to register their automation components into the extensibility object model.

Syntax

object.**AddIns**

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

Visual Basic: DataGrid Control

Visual Studio 6.0

AddNewMode Property

See Also Example Applies To

Returns a value that describes the location of the current cell with respect to the grid's AddNew row. Read-only at run time and not available at design time.

Syntax

object.**AddNewMode**

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

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

Values

The **AddNewMode** property returns one of the following:

Constant	Value	Description
dbgNoAddNew	0	The current cell is not in the last row, and no **AddNew** operation is pending.
dbgAddNewCurrent	1	The current cell is in the last row, but no **AddNew** operation is pending.
dbgAddNewPending	2	The current cell is in the next to last row as a result of a pending **AddNew** operation initiated by the user through the grid's user interface, or by code as a result of setting the **Value** or **Text** properties of a column.

Remarks

If the **AllowAddNew** property is **True**, the last row displayed in the grid is left blank to permit users to enter new records. If the **AllowAddNew** property is **False**, the blank row is not displayed, and **AddNewMode** always returns 0.

This documentation is archived and is not being maintained.

Visual Basic: MAPI Controls

Visual Studio 6.0

AddressCaption Property

See Also Example Applies To

Specifies the caption appearing at the top of the Address Book dialog box when the **Show** method is specified with the *value* argument missing or set to **False**.

Syntax

```
object.AddressCaption [ = value ]
```

The **AddressCaption** 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 address book dialog box caption.

Remarks

If this property is a null or empty string, the default value of the Address Book is used.

Data Type

String

Visual Basic: MAPI Controls

Visual Studio 6.0

AddressEditFieldCount Property

See Also Example Applies To

Specifies which edit controls to display to the user in the Address Book dialog box when the **Show** method is specified with the *value* argument missing or set to **False**.

Syntax

```
object.AddressEditFieldCount [ = value ]
```

The **AddressEditFieldCount** 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 which edit controls to display, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
0	No edit controls; only browsing is allowed.
1	(Default) Only the To edit control should be present in the dialog box.
2	The To and CC (copy) edit controls should be present in the dialog box.
3	The To, CC (copy), and BCC (blind copy) edit controls should be present in the dialog box.
4	Only those edit controls supported by the messaging system should be present in the dialog box.

Remarks

For example, if **AddressEditFieldCount** is 3, the user can select from the To, CC, and BCC edit controls in the Address Book dialog box. The **AddressEditFieldCount** is adjusted so that it is equal to at least the minimum number of edit controls required by the recipient set.

Data Type

Integer

Visual Basic: MAPI Controls

Visual Studio 6.0

AddressLabel Property

See Also Example Applies To

Specifies the appearance of the To edit control in the Address Book when the **Show** method is specified with the *value* argument missing or set to **False**.

Syntax

```
object.AddressLabel [ = value ]
```

The **AddressLabel** 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 an address label.

Remarks

This property is normally ignored and should contain an empty string to use the default label "To." However, when the **AddressEditFieldCount** property is set to 1, the user has the option of explicitly specifying another label (providing the number of editing controls required by the recipient set equals 1).

Data Type

String

Visual Basic: MAPI Controls

Visual Studio 6.0

AddressModifiable Property

See Also Example Applies To

Specifies whether the Address Book can be modified.

Syntax

object.**AddressModifiable** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A boolean expression specifying whether the Address Book can be modified, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	The user is allowed to modify their personal address book.
False	(Default) The user is not allowed to modify their personal address book.

Data Type

Boolean

Visual Basic: MAPI Controls

Visual Studio 6.0

AddressResolveUI Property

See Also Example Applies To

Specifies whether a dialog box is displayed for recipient name resolution during addressing when the **ResolveName** method is specified.

Syntax

object.**AddressResolveUI** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A boolean expression specifying whether a dialog box is displayed, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	A dialog box is displayed with names that closely match the intended recipient's name.
False	(Default) No dialog box is displayed for ambiguous names. An error occurs if no potential matches are found (no matches is not an ambiguous situation).

Data Type

Boolean

Visual Basic Reference

Visual Studio 6.0

AggregateOn Property

See Also Example Applies To

Returns or sets the name of the child **DECommand** object on which the aggregation is based. To use this property, the **AggregateType** must be set to **deRelation**.

Note You must use a child **DECommand** object when creating aggregates in the Data Environment.

Syntax

object.**AggregateOn** [=*string*]

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

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
string	A string expression that defines the name of the DECommand object on which the aggregation is based.

Visual Basic Reference

Visual Studio 6.0

AggregateType Property

See Also Example Applies To

Returns or sets the type of aggregation.

Syntax

object.**AggregateType** [=*value*]

The **AggregateType** 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 type of aggregate, as described in Settings.

Settings

The settings for *value* are:

Constant	Description
deGrouping	Grouping. This is only available if the **DECommand** object is grouped. The **FieldToAggregate** item can be any field in the current **DECommand** object. This is because grouping a **DECommand** object creates two Recordsets: one contains the grouped fields and the other contains the non-grouped fields.
deRelation	Relation. The **FieldToAggregate** item can be any field of the **DECommand** object that is specified in the **AggregateOn** property.
deGrandTotal	Grand Total. This is only available if the **DECommand** object is the top-most Command in a hierarchy. If the **DECommand** object is grouped, only the grouping fields are available. If ungrouped, all fields are available.

Visual Basic Reference

Visual Studio 6.0

Align Property

See Also Example Applies To

Returns or sets a value that determines whether an object is displayed in any size anywhere on a form or whether it's displayed at the top, bottom, left, or right of the form and is automatically sized to fit the form's width.

Syntax

object.**Align** [= *number*]

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

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Number	An integer that specifies how an object is displayed, as described in Settings.

Settings

The settings for *number* are:

Constant	Setting	Description
VbAlignNone	0	(Default in a non-MDI form) None size and location can be set at design time or in code. This setting is ignored if the object is on an MDI form.
VbAlignTop	1	(Default in an MDI form) Top object is at the top of the form, and its width is equal to the form's **ScaleWidth** property setting.
VbAlignBottom	2	Bottom object is at the bottom of the form, and its width is equal to the form's **ScaleWidth** property setting.
VbAlignLeft	3	Left object is at the left of the form, and its width is equal to the form's **ScaleWidth** property setting.
VbAlignRight	4	Right object is at the right of the form, and its width is equal to the form's **ScaleWidth** property setting.

Remarks

You can use the **Align** property to quickly create a toolbar or status bar at the top or bottom of a form. As a user changes the size of the form, an object with **Align** set to 1 or 2 automatically resizes to fit the width of the form.

PictureBox and **Data** controls are the only standard controls that can be placed on an MDI form. The internal area of an MDI form is defined by the space not covered by controls. When an MDI child form is maximized within the parent MDI form, it won't cover any controls.

Use *number* settings 3 and 4 to align toolbars at the left and right sides of a form or MDI form. If there are two toolbars in a corner of an MDI form, the top- or bottom-aligned one extends to the corner, taking precedence over the left- or right-aligned one. Left- and right-aligned objects occupy the internal area on an MDI form, just like top- and bottom-aligned objects.

Visual Basic Reference

Align Property Example

This example uses a **PictureBox** control as a toolbar on an **MDIForm** object, with a **CommandButton** control to move the **PictureBox** from the top to the bottom of the form. To try this example, create a new **MDIForm** and set the **MDIChild** property of Form1 to **True**. Draw a **PictureBox** on the **MDIForm**, and put a **CommandButton** on the **PictureBox**. Paste the code into the Declarations section of the **MDIForm**, and then press F5. Click the **CommandButton** to move the **PictureBox**.

```
Private Sub Command1_Click ()
   If Picture1.Align = vbAlignTop Then
      Picture1.Align = vbAlignBottom
  ' Align to bottom of form.
   Else
      Picture1.Align = vbAlignTop
  ' Align to top of form.
   End If
End Sub
```

Visual Basic Reference

Visual Studio 6.0

Alignable Property

See Also Example Applies To

Returns or sets a value determining if a control is alignable, and can use the extender **Align** property. The **Alignable** property is read/write at the controls authoring time, and not available at run time.

Settings

The settings for **Alignable** are:

Setting	Description
True	The control is alignable; the container will add the **Align** property to the extender object.
False	(Default) The control is not alignable.

Remarks

The alignment of the control itself will be handled by the container; the author of the control can use the **Align** extender property to decide how to redraw the control and arrange the constituent controls in response to an alignment.

Note Not all containers support alignable controls. Error trapping should be used if you access the **Align** extender property to determine how your control has been aligned.

Visual Basic: Windows Controls

Visual Studio 6.0

Alignment Property (ColumnHeader Object)

See Also Example Applies To

Returns or sets the alignment of text in a **ColumnHeader** object.

Syntax

object.**Alignment** [= *integer*]

The **Alignment** Property syntax has these parts:

Part	Description
object	An object expression that evaluates to a **ColumnHeader** object.
integer	An integer that determines the alignment, as described in Settings.

Settings

The settings for *integer* are:

Constant	Value	Description
lvwColumnLeft	0	(Default) Left. Text is aligned left.
lvwColumnRight	1	Right. Text is aligned right.
lvwColumnCenter	2	Center. Text is centered.

Visual Basic: Windows Controls

Alignment Property (Panel Object) Example

This example adds two **Panel** objects to a **StatusBar** control and aligns the text in each panel using one of the three available styles. To try the example, place a **StatusBar** control on a form and paste the code into the Declarations section of the form. Run the example.

```
Private Sub Form_Load()
  ' Declare variables.
  Dim pnlX As Panel
  Dim I As Integer

  For I = 1 To 2  ' Add two panels.
  StatusBar1.Panels.Add
  Next I

  For I = 1 To 3  ' Add pictures to each Panel.
  Set pnlX = StatusBar1.Panels(I)
  Set pnlX.Picture = LoadPicture("Graphics\icons\comm\net12.ico")
      ' Set AutoSize and MinWidth so that panels
      ' are always in view.
  pnlX.AutoSize = sbrSpring
  pnlX.MinWidth = 1
  Next I

  ' Set styles and alignment.
  With StatusBar1.Panels
  .Item(1).Text = "Left"
  .Item(1).Alignment = sbrLeft ' Left alignment.
  .Item(2).Text = "Center"
  .Item(2).Alignment = sbrCenter ' Centered alignment.
  .Item(3).Text = "Right"
  .Item(3).Alignment = sbrRight ' Right alignment.
  End With
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

Alignment Property (Panel Object)

See Also Example Applies To

Returns or sets the alignment of text in the caption of an object.

Syntax

object.**Alignment** [= *number*]

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

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

Settings

The settings for the Panel object *number* are:

Constant	Value	Description
sbrLeft **hdrLeft**	0	(Default). Text appears left-justified and to right of bitmap.
sbrCenter **hdrCenter**	1	Text appears centered and to right of bitmap.
sbrRight **hdrRight**	2	Text appears right-justified and to left of bitmap.

Remarks

As well as positioning the text, the **Alignment** property specifies the position of the bitmap, as described in Settings. There is no way to independently position the bitmap within the panel.

Visual Basic: Windows Controls

Visual Studio 6.0

Alignment Property (UpDown Control)

See Also Example Applies To

Returns or sets a value that determines the alignment of the **UpDown** control with its buddy control.

Syntax

object.**Alignment** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A value that specifies the alignment of the **UpDown** control with its buddy control, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
cc2alignmentLeft	0	The **UpDown** control is aligned to the left of its buddy control.
cc2alignmentRight	1	(Default). The **UpDown** control is aligned to the right of its buddy control.

Remarks

Use the **Alignment** property to specify the positioning of the **UpDown** control next to its buddy control. By default, the **UpDown** control is displayed on the right side of the buddy control.

Setting the **Alignment** property automatically realigns the **UpDown** control with its buddy control. The buddy control's width is reduced by the width of the UpDown control, so that the overall width of the two controls is the same as the buddy control was alone.

Note The **Alignment** property ignores the **Orientation** property when aligning to the **UpDown** control.

Visual Basic Reference

Visual Studio 6.0

Alignment Property

See Also Example Applies To

Returns or sets a value that determines the alignment of a **CheckBox** or **OptionButton** control, text in a control, or values in a column of a **DataGrid** control. Read-only at run time for **CheckBox**, **OptionButton**, and **TextBox** controls.

Syntax

object.**Alignment** [= *number*]

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

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Number	An integer that specifies the type of alignment, as described in Settings.

Settings

For **CheckBox** and **OptionButton** controls, the settings for *number* are:

Constant	Setting	Description
VbLeftJustify	0	(Default) Text is left-aligned; control is right-aligned.
VbRightJustify	1	Text is right-aligned; control is left-aligned.

For **Label** and **TextBox** controls, the settings for *number* are:

Constant	Setting	Description
VbLeftJustify	0	(Default) Text is left-aligned.
VbRightJustify	1	Text is right-aligned.
VbCenter	2	Text is centered.

For a **DataGrid** column, the settings for *number* are:

Constant	Setting	Description
DbgLeft	0	Text is left-aligned.
DbgRight	1	Text is right-aligned.
DbgCenter	2	Text is centered.
DbgGeneral	3	(Default) General Text is left-aligned; numbers are right-aligned.

Remarks

You can display text to the right or left of **OptionButton** and **CheckBox** controls. By default, text is left-aligned.

The **MultiLine** property in a **Textbox** control must be set to **True** for the **Alignment** property to work correctly. If the **MultiLine** property setting of a **TextBox** control is **False**, the **Alignment** property is ignored.

Visual Basic: DataGrid Control

Visual Studio 6.0

AllowAddNew Property

See Also Example Applies To

Returns or sets a value indicating whether the user can add new records to the **Recordset** object underlying a **DataGrid** control.

Syntax

object.**AllowAddNew** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines whether a user can add new records, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	Users can add records to the **Recordset** object underlying the **DataGrid** control.
False	Users can't add records to the **Recordset** underlying the **DataGrid** control.

Remarks

If the **AllowAddnew** property is **True**, the last row displayed in the **DataGrid** control is left blank to permit users to enter new records. If the **AllowAddNew** property is **False**, no blank line is displayed.

The underlying **Recordset** may not enable insertions even if the **AllowAddNew** property is **True**. In this case, an error occurs when the user tries to add a record.

Visual Basic: DataGrid Control

AllowAddNew, AllowDelete, AllowUpdate Properties Example

This example checks the value of a check box. If it is **False**, the user can't make changes to the grid.

```
Private Sub Form_Load ()
   If Check1.Value = 0 Then
      DataGrid1.AllowDelete = False
      DataGrid1.AllowAddNew = False
      DataGrid1.AllowUpdate = False
   End If
End Sub
```

Visual Basic: DataGrid Control

Visual Studio 6.0

AllowArrows Property

See Also Example Applies To

Sets or returns a value that determines whether the control uses the arrow keys for grid navigation.

Syntax

object.**AllowArrows** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines the arrow keys are used for grid navigation, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	(Default) The user can use the arrow keys to move both from cell to cell and row to row.
False	The left and right arrow keys will move focus from control to control and cannot be used to move between cells.

Remarks

The user cannot use the arrow keys to move out of the **DataGrid** control when this property is set to **True**. If the **WrapCellPointer** property is also set to **True**, then the arrow keys will wrap around rows and the user can navigate the entire grid using the arrow keys.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

AllowBigSelection Property

See Also Example Applies To

Returns or sets a value that determines whether clicking on a column or row header should cause the entire column or row to be selected.

Syntax

object.**AllowBigSelection** [=*Boolean*]

The **AllowBigSelection** 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 entire column or row is selected when the header is clicked.

Settings

The settings for *Boolean* are:

Setting	Description
True	Default. When the user clicks the header, the entire column or row is selected.
False	When the user clicks the header, only the header is selected.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

AllowBigSelection Property Example

The following code example allows an entire column or row to be selected when the user clicks on the header.

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

```
Sub Form1_Load ()
   MSHFlexGrid1.AllowBigSelection =True
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

AllowColumnReorder Property

See Also Example Applies To

Returns or sets a value that determines if the user can reorder columns using the mouse.

Syntax

object.**AllowColumnReorder** [= *boolean*]

The **AllowColumnReorder** 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 user can reorder columns, as shown in Settings.

Settings

The settings for *boolean* are:

Constant	Description
False	(Default) User can't reorder columns.
True	Users can reorder columns.

Visual Basic: Windows Controls

Visual Studio 6.0

AllowCustomize Property

See Also Example Applies To

Returns or sets a value determining if a **Toolbar** control can be customized by the end user with the Customize Toolbar dialog box.

Syntax

object.**AllowCustomize** [= *boolean*]

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

Part	**Description**
object	An object expression that evaluates to a **Toolbar** control.
boolean	A constant or value that determines if the user can customize a **Toolbar** control, as described in Settings.

Settings

The settings for *boolean* are:

Setting	**Description**
True	Allows the end user to invoke the Customize Toolbar dialog box by double clicking a **Toolbar** control.
False	Customization of the **Toolbar** control with the Customize Toolbar dialog box is not allowed.

Remarks

If the **AllowCustomize** property is set to **True**, double-clicking a **Toolbar** control at run time invokes the Customize Toolbar dialog box.

The Customize Toolbar can also be invoked with the **Customize** method.

Visual Basic: DataGrid Control

Visual Studio 6.0

AllowDelete Property

See Also Example Applies To

Returns or sets a value indicating whether the user can delete records from the **Recordset** object underlying a **DataGrid** control.

Syntax

object.**AllowDelete** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines whether a user can delete records, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	Users can delete records from the **Recordset** object underlying the **DataGrid** control.
False	Users can't delete records from the **Recordset** underlying the **DataGrid** control.

Remarks

Use the **AllowDelete** property to prevent the user from deleting records from the **Recordset** through interaction with the **DataGrid** control.

The underlying **Recordset** may not enable deletions even if the **AllowDelete** property is **True** for the **DataGrid** control. In this case, an error occurs when the user tries to delete a record.

Note After deleting a record from a **DataGrid** control, you should use the **Refresh** method on it to force the **DataGrid** to update. Otherwise, although a record has been deleted from the underlying recordset, the record will still show in the **DataGrid**.

Visual Studio 6.0

Visual Basic: MSChart Control

AllowDithering Property

See Also Example Applies To

Returns or sets a value that determines whether to disable color dithering for charts on 8-bit color monitors in order to enable use of **MSChart** control's own color palette and enhance the chart display.

Syntax

object.**AllowDithering** [=*boolean*]

The **AllowDithering** 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 a color dithering is allowed, as described in Settings.

Settings

The settings for *boolean* are:

Setting	**Description**
True	Color dithering is allowed.
False	(Default) MSChart control's color palette is used for enhanced color matching and display.

Visual Studio 6.0

Visual Basic: MSChart Control

AllowDynamicRotation Property

See Also Example Applies To

Returns or sets a value that indicates whether users can interactively rotate three-dimensional charts by holding down the control key to display the rotation cursor.

Syntax

object.**AllowDynamicRotation** [= *boolean*]

The **AllowDynamicRotation** 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 a dynamic rotation is allowed, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) The user can interactively rotate the chart with the cursor.
False	The user cannot interactively rotate the chart with the cursor.

Visual Basic: DataGrid Control

Visual Studio 6.0

AllowFocus Property

See Also Example Applies To

Sets or returns a value that determines whether cells within a split can receive focus.

Syntax

object.**AllowFocus** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines whether a cell receives focus, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	(Default) The user will be able to interactively select the split, giving it focus.
False	The user will not be able to interactively select the split. When clicked on, the split will not receive focus and the control that previously had focus will retain it.

Remarks

Use this property in combination with the **AllowSizing** property to completely prohibit the user from making any changes to a split (by setting both properties to **False**). Unselectable splits are passed over when **TabAcrossSplits** is set to **True**.

Visual Basic: MaskedEdit Control

Visual Studio 6.0

AllowPrompt Property

See Also Example Applies To

Determines whether or not the prompt character is a valid input character.

Syntax

[*form*.]**MaskedEdit**.**AllowPrompt** [= {**True** | **False**}]

Remarks

The **AllowPrompt** property settings are as follows:

Setting	Description
False	(Default) The prompt character is not a valid input character. A ValidationError event is triggered if you enter the prompt character.
True	The prompt character is a valid input character.

For example, suppose you have defined a prompt character of 0, and you want the **Masked Edit** control to accept five digits between 0 and 9. You specify a mask of #####. If the **AllowPrompt** property is **False** and you enter 0, a ValidationError event occurs. If **AllowPrompt** is set to **True**, you can enter 0 as a valid input character.

Data Type

Integer (Boolean)

Visual Basic: DataGrid Control

Visual Studio 6.0

AllowRowSizing Property

See Also Example Applies To

Returns or sets a value indicating whether a user can resize the rows of the **DataGrid** control or **Split** object at run-time.

Syntax

object.**AllowRowSizing** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines whether a user can resize rows, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	Rows can be sized by the user.
False	Rows can't be sized by the user.

Remarks

If the **AllowRowSizing** property is **True**, the mouse pointer turns into a double-headed (Size N S) arrow when positioned over the row divider between any record selectors, and the user can resize the rows by dragging. Any change in row size causes a RowResize event.

All rows of the **DataGrid** control are always the same height, which is determined by the **RowHeight** property.

Note Even if the **AllowRowSizing** property is **False**, the height of the rows can still be changed programmatically with the **RowHeight** property.

Visual Basic: DataGrid Control

AllowRowSizing Property Example

This example checks the database to see if it has any memo fields; if not, row resizing is disabled.

```
Sub CheckForMemoField()
   Dim Fld As Field
   DataGrid1.AllowRowSizing = False
   For Each Fld in Data1.Recordset.Fields
      If Fld.Type = dbMemo Then
         DataGrid1.AllowRowSizing = True
         DataGrid1.RowHeight = DataGrid1.RowHeight * 2
         Exit For
      End If
   Next
End Sub
```

Visual Studio 6.0

Visual Basic: MSChart Control

AllowSelections Property

See Also Example Applies To

Returns or sets a value that indicates whether or not users can select chart objects.

Syntax

object.**AllowSelections** [= *boolean*]

The **AllowSelections** 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 selections can be made, as described in Settings.

Settings

The settings for *boolean* are:

Setting	**Description**
True	(Default) The user can interactively select chart objects.
False	The user cannot select chart objects.

Visual Studio 6.0

Visual Basic: MSChart Control

AllowSeriesSelection Property

See Also Example Applies To

Returns or sets a value that indicates whether a series is selected when a user clicks on an individual chart data point.

Syntax

object.**AllowSeriesSelection** [= *boolean*]

The **AllowSeriesSelection** 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 series are selected, as described in Settings.

Settings

The settings for *boolean* are:

Setting	**Description**
True	(Default) Users can select a series by clicking a data point.
False	Clicking a data point selects only that data point, not the entire series.

Visual Basic: DataGrid Control

Visual Studio 6.0

AllowSizing Property

See Also Example Applies To

Returns or sets a value indicating whether a user can resize columns or splits in the **DataGrid** control at run-time.

Syntax

object.**AllowSizing** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines whether a column or split can be resized, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	(Default for **Column**) User can resize column or split.
False	(Default for **Split**) User can't resize column or split.

Remarks

If the **AllowSizing** property is **True**, the mouse pointer turns into a double-headed (Size W E) arrow when positioned over the divider of the specified column, and the user can resize the column by dragging. Any change in column size causes a ColResize event.

For the leftmost split with **AllowSizing** set to **True**, the mouse pointer turns into a pair of vertical lines with a downward arrow when positioned over that split's size box (at the lower left corner), and the user can create a new split by dragging. The creation of a new split causes a SplitChange event.

If **AllowSizing** is **True** for any other split, the mouse pointer turns into a pair of vertical lines with a double-headed arrow when positioned over that split's size box, and the user can resize the split by dragging. No event is fired in this case (except for the standard mouse events).

Visual Basic: DataGrid Control

AllowSizing Property Example

This example prevents the user from resizing or editing the first three columns of the grid.

```
Private Sub Form_Load ()
   Dim I
   For I = 0 to 2
      DataGrid1.Columns(I).AllowSizing = False
      DataGrid1.Columns(I).Locked = True
   Next I
End Sub
```

Visual Basic: DataGrid Control

Visual Studio 6.0

AllowUpdate Property

See Also Example Applies To

Returns or sets a value indicating whether a user can modify any data in the **DataGrid** control.

Syntax

object.**AllowUpdate** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines whether the user can change data, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	The user can modify data in the **DataGrid** control
False	The user can't modify data in the **DataGrid** control

Remarks

When the **AllowUpdate** property is **False**, the user can still scroll through the **DataGrid** control and select data, but can't change any of the values; any attempt to change the data in the grid is ignored.

You can also use the **Column** object properties to make individual columns of the **DataGrid** control read-only, but the **AllowUpdate** property setting takes precedence over the column settings (without changing the column settings).

Note The **Recordset** object may not enable updates even if **AllowUpdate** is **True** for the **DataGrid** control; in this case a trappable error occurs when the user tries to change the record.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

AllowUserResizing Property

See Also Example Applies To

Returns or sets a value that determines whether the user can use the mouse to resize rows and columns in the **MSHFlexGrid**.

Syntax

object.**AllowUserResizing** [=*value*]

The **AllowUserResizing** 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 whether a user can resize rows and columns, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
flexResizeNone	0	None. Default. The user cannot resize with the mouse.
flexResizeColumns	1	Columns. The user can resize columns using the mouse.
flexResizeRows	2	Rows. The user can resize rows using the mouse.
flexResizeBoth	3	Both. The user can resize columns and rows using the mouse.

Remarks

To resize rows or columns, the mouse must be over the fixed area of the **MSHFlexGrid** and close to a border between rows and columns. The mouse pointer changes into an appropriate sizing pointer, and the user can drag the row or column to change the row height or column width.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

AllowUserResizing Property Example

The following code example adds user resizing functionality.

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

```
Sub Form1_Load ()
   MSHFlexGrid1.AllowUserResizing =True
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

AllowVertical Property (Band Object)

See Also Example Applies To

Returns or sets a value indicating whether a band will be displayed when the **CoolBar** controls orientation is set to vertical.

Syntax

object.**AllowVertical** [= *boolean*]

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

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

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Band is visible.
False	Band is hidden.

Remarks

To hide a band at startup when the **Orientation** is set to Vertical, set the **AllowVertical** property to **False** at design time. Setting this property in code enables you to hide and later redisplay a band at run time in response to a particular event.

Note When the **Orientation** property is set to vertical, both **AllowVertical** and **Visible** must be **True** for the band to be visible. If either is **False**, the band will not be visible.

Visual Basic: RDO Data Control

Visual Studio 6.0

AllowZeroLength Property (Remote Data)

See Also Example Applies To

Returns a value that indicates whether a zero-length string ("") is a valid setting for the **Value** property of an **rdoColumn** object with a data type of **rdTypeCHAR**, **rdTypeVARCHAR**, or **rdTypeLONGVARCHAR**.

Syntax

object.**AllowZeroLength**

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

Return Values

The **AllowZeroLength** property return values are:

Value	Description
True	A zero-length string is a valid value.
False	A zero-length string isn't a valid value.

Remarks

If **AllowZeroLength** is **False** for a column, you must use Null to represent "unknown" states you cannot use empty strings.

Visual Basic Reference

Visual Studio 6.0

Ambient Property

See Also Example Applies To

Returns an **AmbientProperties** object holding the ambient properties of the container. The **Ambient** property is not available at the controls authoring time, and read-only at the controls run time.

Syntax

*object***.Ambient**

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

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

Visual Studio 6.0

Visual Basic: MSChart Control

AmbientIntensity Property

See Also Example Applies To

Returns or sets the percentage of ambient light illuminating a three-dimensional chart.

Syntax

object.**AmbientIntensity** [= *intensity*]

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

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
intensity	Single. The chart light intensity. Valid values are 0 to 1. If set to 1, all sides of the chart elements are fully illuminated no matter what light sources are turned on. If set at 0, there is no contribution from ambient light; only the sides of the chart elements facing active light sources are illuminated.

Visual Studio 6.0

Visual Basic: MSChart Control

AngleUnit Property

See Also Example Applies To

Returns or sets the unit of measure used for all chart angles.

Syntax

object.**AngleUnit** [= *unit*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
unit	Integer. A VtAngleUnits constant describing the unit of measure. The angles can be measured in degrees, radians, or grads.

Visual Basic Reference

Visual Studio 6.0

App Property

See Also Example Applies To

Returns the **App** object, a global object accessed with the **App** keyword. It determines or specifies information about the application's title, version information, the path and name of its executable file and Help files, and whether or not a previous instance of the application is running.

Syntax

App

Remarks

The **App** object has no events or methods.

Visual Basic Reference

Visual Studio 6.0

Appearance Property

See Also Example Applies To

Returns or sets the paint style of controls on an **MDIForm** or **Form** object at design time. Read-only at run time.

Syntax

object.**Appearance**

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

Settings

The **Appearance** property settings are:

Setting	Description
0	Flat. Paints controls and forms without visual effects.
1	(Default) 3D. Paints controls with three-dimensional effects.

Remarks

If set to 1 at design time, the **Appearance** property draws controls with three-dimensional effects. If the form's **BorderStyle** property is set to Fixed Double (**vbFixedDouble**, or 3), the caption and border of the form are also painted with three-dimensional effects. Setting the **Appearance** property to 1 also causes the form and its controls to have their **BackColor** property set to the color selected for 3D Objects in the Appearance tab of the operating system's Display Properties dialog box.

Setting the **Appearance** property to 1 for an **MDIForm** object affects only the MDI parent form. To have three-dimensional effects on MDI child forms, you must set each child form's **Appearance** property to 1.

Visual Basic Reference

Visual Studio 6.0

Appearance Property (ActiveX Controls)

See Also Example Applies To

Returns or sets a value that determines the appearance of the object.

Syntax

object.**Appearance**

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

Settings

The **Appearance** property settings are:

Setting	Description
0	Flat. Paints controls and forms without visual effects.
1	(Default) 3D. Paints control or object with three-dimensional effects.

Visual Basic: Windows Controls

Visual Studio 6.0

Appearance Property (FlatScrollBar Control)

See Also Example Applies To

Returns or sets the appearance of a **FlatScrollBar** control

Syntax

object.**Appearance** [= *integer*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
integer	A numeric expression specifying the style of the scroll bar, as described in Settings.

Settings

The settings for *integer* are:

Constant	Value	Description
fsb3D	0	Scroll bar will have the three-dimensional appearance of a standard Windows scroll bar.
fsbFlat	1	(Default) Scroll bar will appear two-dimensional.
fsbTrack3D	2	Scroll bar will appear two-dimensional, with the thumb and arrow buttons becoming three-dimensional when the mouse passes over them.

Remarks

The **Appearance** property sets the look of the scroll bar, and determines whether it will be 2-D, 3-D, or a combination. The combination scroll bar changes the scroll arrows and thumb from 2-D to 3-D in response to the mouse pointer, and lets you create a dynamic interface.

Visual Basic Reference

Visual Studio 6.0

AppIsRunning Property

See Also Example Applies To

Returns or sets a value that indicates whether the application that created the object in the **OLE** container control is running. Not available at design time.

Syntax

object.**AppIsRunning** [= *boolean*]

The **AppIsRunning** 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 whether or not the application that produced the object in the **OLE** container control is running, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The application that produced the object in the **OLE** container control is running.
False	The application that produced the object in the **OLE** container control isn't running.

Remarks

You can set the value of the **AppIsRunning** property to start the application that produces the object in the **OLE** container control. Doing this causes objects to activate more rapidly. You can also set this property to **False** to close the application when the object loses the focus.

Visual Basic Reference

Visual Studio 6.0

Application Property (WebClass)

See Also Example Applies To

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

Syntax

object.**Application**

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

Remarks

A WebClass uses the **Application** object to manage state that is shared across multiple **WebClass** object instances.

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

Visual Basic: DataGrid Control

Visual Studio 6.0

ApproxCount Property

See Also Example Applies To

Returns the approximate number of rows in the grid.

Syntax

object.**ApproxCount**

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

Remarks

This property returns the approximate row count used by the grid to calibrate the vertical scroll bar.

Typically, the **ApproxCount** property is used to improve the accuracy of the vertical scroll bar. This is particularly useful for situations where the number of rows is known in advance, such as when a grid is used in conjunction with an array.

Note Getting the **ApproxCount** property will query the underlying data source.

Visual Basic Reference

Visual Studio 6.0

Archive, Hidden, Normal, System Properties

See Also Example Applies To

Return or set a value that determines whether a **FileListBox** control displays files with Archive, Hidden, Normal, or System attributes.

Syntax

object.**Archive** [= *boolean*]

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

object.**Normal** [= *boolean*]

object.**System** [= *boolean*]

The **Archive**, **Hidden**, **Normal**, and **System** property syntaxes have these parts:

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Boolean	A Boolean expression that specifies the type of files displayed, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default for Archive and Normal) Displays files with the property's attribute in the **FileListBox** control.
False	(Default for Hidden and System) Displays files without the property's attribute in the **FileListBox** control.

Remarks

Use these properties to specify the type of files to display in a **FileListBox** control, based on standard file attributes used in the operating environment. Setting any of these properties with code at run time resets the **FileListBox** control to display only those files with the specified attributes.

For example, in a find-and-replace operation you could display only system files by setting the **System** property to **True** and the other properties to **False**. Or, as part of a file backup procedure, you could set the **Archive** property to **True** to list only those files modified since the previous backup.

Visual Basic: Windows Controls

Visual Studio 6.0

Arrange Property (ListView Control)

See Also Example Applies To

Returns or sets a value that determines how the icons in a **ListView** control's Icon or SmallIcon view are arranged.

Syntax

```
object.Arrange [= value]
```

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

Part	Description
object	An object expression that evaluates to a **ListView** control.
value	An integer or constant that determines how the icons or small icons are arranged, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
lvwNone	0	(Default) None.
lvwAutoLeft	1	Left. Items are aligned automatically along the left side of the control.
lvwAutoTop	2	Top. Items are aligned automatically along the top of the control.

Visual Basic: Windows Controls

Arrange Property Example

This example adds several **ListItem** objects and subitems to a **ListView** control. When you click on an **OptionButton** control, the **Arrange** property is set with the **Index** value of the **OptionButton**. To try the example, place a control array of three **OptionButton** controls, a **ListView** control, and two **ImageList** controls on a form and paste the code into the form's Declarations section. Run the example and click on an **OptionButton** to change the **Arrange** property.

```
Private Sub Option1_Click(Index as Integer)
   ' Set Arrange property to Option1.Index.
   ListView1.Arrange = Index
End Sub

Private Sub Form_Load()
   ' Label OptionButton controls with Arrange choices.
      Option1(0).Caption = "No Arrange"
      Option1(1).Caption = "Align Auto Left"
      Option1(2).Caption = "Align Auto Top"

   ' Declare variables for creating ListView and ImageList objects.
   Dim i As Integer
   Dim itmX As ListItem    ' Object variable for ListItems.
   Dim imgX As ListImage    ' Object variable for ListImages.

   ' Add a ListImage object to an ImageList control.
   Set imgX = ImageList1.ListImages. _
   Add(,,LoadPicture("icons\mail\mail01a.ico"))

   ListView1.Icons = ImageList1    ' Associate an ImageList control.

   ' Add ten ListItem objects, each with an Icon.
   For i = 1 To 10
      Set itmX = ListView1.ListItems.Add()
      itmX.Icon = 1    ' Icon.
      itmX.Text = "ListItem " & i
   Next i
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

Arrows Property

See Also Example Applies To

Returns or sets a value that determines which scroll arrows will be enabled.

Syntax

object.**Arrows** [= *integer*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
integer	A numeric expression specifying the enabled scroll button or buttons, as described in Settings.

Settings

The settings for *integer* are:

Constant	Value	Description
cc2Both	0	(Default) Both the left and right (or up and down, depending on **Orientation**) scroll arrows will be enabled.
cc2LeftUp	1	Only the left (or up) scroll button will be enabled.
cc2RightDown	2	Only the right (or down) scroll button will be enabled.

Remarks

When the **Orientation** of the **FlatScrollBar** changes from horizontal to vertical, the left scroll arrow becomes the up arrow, and the right scroll arrow becomes the down arrow.

The most common use of the **Arrows** property is to disable the appropriate scroll arrow when the maximum or minimum value of the control is reached.

Visual Basic Reference

Visual Studio 6.0

AsyncCount Property

See Also Example Applies To

Returns the number of asynchronous operations still executing.

Syntax

object.**AsyncCount**

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

Return Type

Integer

Remarks

Query the **AsyncCount** property before closing the Data Report designer to determine the number of operations still executing. You may want to cancel closing the designer until all operations have finished.

Visual Basic Reference

AsyncCount Property Example

The example invokes the **ExportReport** method and queries the **AsyncCount** property in a **While** loop to determine if any asynchronous operations are still executing. When all are done, the **DataReport** object is unloaded.

```
DataReport1.ExportReport rptKeyText, "c:\MyDocuments\Report", True
While DataReport1.AsyncCount > 0
   DoEvents
Wend
Unload DataReport1
```

Visual Basic: RDO Data Control

Visual Studio 6.0

AsyncCheckInterval Property (Remote Data)

See Also Example Applies To

Returns or sets a value specifying the number of milliseconds that RDO waits between checks to see if an asynchronous query is complete.

Syntax

object.**AsyncCheckInterval** [= *value*]

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

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

Remarks

When you use the **rdAsyncEnable** option to execute a query asynchronously, RDO polls the data source periodically to determine if the query has completed. You can change the duration of time between checks by using the **AsyncCheckInterval** property. RDO also checks the status of an asynchronous query when you examine the **StillExecuting** property.

The **AsyncCheckInterval** property defaults to 1000 milliseconds (once a second).

Polling too often can adversely affect both server and workstation performance. Polling less frequently can improve performance, but may affect how quickly data is made available to the user.

As long as the asynchronous query is executing, the **StillExecuting** property returns **True**. Once the query is completed, the **StillExecuting** property is set to false and the QueryComplete event is fired. You can also interrupt and end an asynchronous query by using the **rdoResultset** object's **Cancel** or **Close** method.

Visual Basic: Page Designer

Visual Studio 6.0

AsyncLoad Property

See Also Example Applies To

Returns or sets a Boolean value that determines whether or not the objects on the page are loaded asynchronously with the execution of the code. Not available at run time.

Remarks

By default, **AsyncLoad** is set to **False**, meaning code execution does not begin until the browser has retrieved all elements on the page. If set to **True**, code execution begins as soon as the browser has downloaded the code, regardless of the download status of other components.

Before changing the value of this property, be sure that your code is prepared to deal with the fact that controls on the page may not be loaded at the time execution begins.

Visual Basic Reference

Visual Studio 6.0

AsyncType Property

See Also Example Applies To

Returns or sets the type of the data returned by the **Value** property. This property is available only as an argument of the **AsyncRead** method.

Syntax

object.**AsyncType** = *dataType*

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
dataType	An integer specifying the data type, as shown in Settings below.

Settings

The settings for *dataType* are:

Constant	Value	Description
vbAsyncTypePicture	0	Default. Picture object.
VbAsyncTypeFile	1	The data is provided in a file created by Visual Basic.
VbAsyncTypeByteArray	2	The data is provided as a byte array that contains the retrieved data.

Visual Basic for Applications Reference

Visual Studio 6.0

AtEndOfLine Property

See Also Example Applies To Specifics

Description

Read-only property that returns **True** if the file pointer immediately precedes the end-of-line marker in a **TextStream** file; **False** if it does not.

Syntax

object.**AtEndOfLine**

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

Remarks

The **AtEndOfLine** property applies only to **TextStream** files that are open for reading; otherwise, an error occurs.

The following code illustrates the use of the **AtEndOfLine** property:

```
Dim fs, a, retstring
Set fs = CreateObject("Scripting.FileSystemObject")
Set a = fs.OpenTextFile("c:\testfile.txt", ForReading, False)
Do While a.AtEndOfLine <> True
    retstring = a.Read(1)
    ...
Loop
a.Close
```

Visual Basic for Applications Reference

Visual Studio 6.0

AtEndOfStream Property

See Also Example Applies To Specifics

Description

Read-only property that returns **True** if the file pointer is at the end of a **TextStream** file; **False** if it is not.

Syntax

object.**AtEndOfStream**

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

Remarks

The **AtEndOfStream** property applies only to **TextStream** files that are open for reading; otherwise, an error occurs.

The following code illustrates the use of the **AtEndOfStream** property:

```
Dim fs, a, retstring
Set fs = CreateObject("Scripting.FileSystemObject")
Set a = fs.OpenTextFile("c:\testfile.txt", ForReading, False)
Do While a.AtEndOfStream <> True
    retstring = a.ReadLine
    ...
Loop
a.Close
```

Visual Basic: MAPI Controls

Visual Studio 6.0

AttachmentCount Property

See Also Example Applies To

Returns the total number of attachments associated with the currently indexed message. This property is not available at design time, and is read-only at run time.

Syntax

object.**AttachmentCount**

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

Remarks

The default value is 0. The value of **AttachmentCount** depends on the number of attachments in the current indexed message.

Note With Microsoft Outlook Express 5 and Netscape, the **AttachmentCount** property always returns zero even if there are some attachments in e-mail messages.

Data Type

Long

Visual Basic: MAPI Controls

Visual Studio 6.0

AttachmentIndex Property

See Also Example Applies To

Sets the currently indexed attachment. This property is not available at design time.

Syntax

```
object.AttachmentIndex [ = value ]
```

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A long expression specifying the currently indexed attachment.

Remarks

Specifies an index number to identify a particular message attachment. The index number in this property determines the values in the **AttachmentName**, **AttachmentPathName**, **AttachmentPosition**, and **AttachmentType** properties. The attachment identified by the **AttachmentIndex** property is called the *currently indexed* attachment. The value of **AttachmentIndex** can range from 0 (the default) to **AttachmentCount** -1.

To add a new attachment, set the **AttachmentIndex** to a value greater than or equal to the current attachment count while in the compose buffer (**MsgIndex** = -1). The **AttachmentCount** property is updated automatically to reflect the implied new number of attachments.

For example, if the current **AttachmentCount** property has the value 3, setting the **AttachmentIndex** property to 4 adds 2 new attachments and increases the **AttachmentCount** property to 5.

To delete an existing attachment, specify the **Delete** method with the *value* parameter set to 2. Attachments can be added or deleted only when the **MsgIndex** property is set to -1.

Data Type

Long

Visual Basic: MAPI Controls

Visual Studio 6.0

AttachmentName Property

See Also Example Applies To

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

Syntax

```
object.AttachmentName [ = value ]
```

The **AttachmentName** 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 name of the currently indexed attachment file.

Remarks

The file name specified is the file name seen by the recipients of the currently indexed message. If **AttachmentName** is an empty string, the file name from the **AttachmentPathName** property is used.

If the attachment is an OLE object, **AttachmentName** contains the class name of the object, for example, "Microsoft Excel Worksheet."

Attachments in the read buffer are deleted when a subsequent fetch occurs. The value of **AttachmentName** depends on the currently indexed message as selected by the **AttachmentIndex** property.

Data Type

String

Visual Basic: MAPI Controls

Visual Studio 6.0

AttachmentPathName Property

See Also Example Applies To

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

Syntax

object.**AttachmentPathName** [= *value*]

The **AttachmentPathName** 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 full path name of the currently indexed attachment.

Remarks

If you attempt to send a message with an empty string for a path name, an error results. Attachments in the read buffer are deleted when a subsequent fetch occurs. Attachments in the compose buffer need to be manually deleted. The value of **AttachmentPathName** depends on the currently indexed message, as selected by the **AttachmentIndex** property.

Data Type

String

Visual Basic: MAPI Controls

Visual Studio 6.0

AttachmentPosition Property

See Also Example Applies To

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

Syntax

object.**AttachmentPosition** [= *value*]

The **AttachmentPosition** property syntax these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A long expression specifying the position of the currently indexed attachment.

Remarks

To determine where an attachment is placed, count the characters in the message body and decide which character position you wish to replace with the attachment. The character count at that position should be used for the **AttachmentPosition** value.

For example, in a message body that is five-characters long, you could place an attachment at the end of the message by setting **AttachmentPosition** equal to 4. (The message body occupies character positions 0 to 4.)

You can't place two attachments in the same position within the same message. In addition, you can't place an attachment beyond the end of the message body.

The value of **AttachmentPosition** depends on the currently indexed attachment, as selected by the **AttachmentIndex** property.

Data Type

Long

Visual Basic: MAPI Controls

Visual Studio 6.0

AttachmentType Property

See Also Example Applies To

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

Syntax

```
object.AttachmentType [ = value ]
```

The **AttachmentType** 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 type of the currently indexed file attachment, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
mapData	0	The attachment is a data file.
mapEOLE	1	The attachment is an embedded OLE object.
mapSOLE	2	The attachment is a static OLE object.

Remarks

The value of **AttachmentType** depends on the currently indexed attachment, as selected by the **AttachmentIndex** property.

Data Type

Integer

Visual Basic for Applications Reference

Visual Studio 6.0

Attributes Property

See Also Example Applies To Specifics

Description

Sets or returns the attributes of files or folders. Read/write or read-only, depending on the attribute.

Syntax

object.**Attributes** [= *newattributes*]

The **Attributes** property has these parts:

Part	Description
object	Required. Always the name of a **File** or **Folder** object.
newattributes	Optional. If provided, *newattributes* is the new value for the attributes of the specified *object*.

Settings

The *newattributes* argument can have any of the following values or any logical combination of the following values:

Constant	Value	Description
Normal	0	Normal file. No attributes are set.
ReadOnly	1	Read-only file. Attribute is read/write.
Hidden	2	Hidden file. Attribute is read/write.
System	4	System file. Attribute is read/write.
Volume	8	Disk drive volume label. Attribute is read-only.
Directory	16	Folder or directory. Attribute is read-only.
Archive	32	File has changed since last backup. Attribute is read/write.
Alias	64	Link or shortcut. Attribute is read-only.
Compressed	128	Compressed file. Attribute is read-only.

Remarks

The following code illustrates the use of the **Attributes** property with a file:

```
Sub SetClearArchiveBit(filespec)
    Dim fs, f, r
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFile(fs.GetFileName(filespec))
    If f.attributes and 32 Then
        r = MsgBox("The Archive bit is set, do you want to clear it?", vbYesNo, "Set/Clear Archive Bit")
        If r = vbYes Then
            f.attributes = f.attributes - 32
            MsgBox "Archive bit is cleared."
        Else
            MsgBox "Archive bit remains set."
        End If
    Else
        r = MsgBox("The Archive bit is not set. Do you want to set it?", vbYesNo, "Set/Clear Archive Bit")
        If r = vbYes Then
            f.attributes = f.attributes + 32
            MsgBox "Archive bit is set."
        Else
            MsgBox "Archive bit remains clear."
        End If
    End If
End Sub
```

Visual Basic Reference

Visual Studio 6.0

Attributes Property (DEDesigner Extensibility)

See Also Example Applies To

When associated with a **DEConnection** object, returns or sets any extra attributes needed for the connection string associated with a **DEConnection** object.

Syntax

object.**Attributes** [=*string*]

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

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
string	A string expression that provides any extra attributes needed.

Visual Basic: RDO Data Control

Visual Studio 6.0

Attributes Property (Remote Data)

See Also Example Applies To

Returns a value that indicates one or more characteristics of an **rdoColumn** object.

Syntax

object.**Attributes**

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

Return Values

The **Attributes** property return value specifies characteristics of the column represented by the **rdoColumn** object and can be a sum of these constants:

Constant	Value	Description
rdFixedColumn	1	The column size is fixed (default for numeric columns) For example, Char, Binary.
rdVariableColumn	2	The column size is variable. For example, VarChar and LongVarChar, VarBinary and LongVarBinary columns.
rdAutoIncrColumn	16	The column value for new rows is automatically incremented to a unique value that can't be changed.
rdUpdatableColumn	32	The column value can be changed.
rdTimeStampColumn	64	The column is a timestamp value. This attribute is set only for **rdClientBatch** cursors.

Remarks

When checking the setting of this property, you should use the **And** operator to test for a specific attribute. Testing for absolute values can jeopardize future compatibility. For example, to determine whether an **rdoColumn** object is fixed-size, you can use code like the following:

```
If MyResultset![ColumnName].Attributes And rdFixedColumn Then...
```

Visual Studio 6.0

Visual Basic: MSChart Control

Auto Property (CategoryScale)

See Also Example Applies To

Returns or sets a value that indicates whether the axis is automatically scaled.

Syntax

object.**Auto** [= *boolean*]

The **Auto** 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 item is displayed, as described in Settings.

Settings

The settings for *boolean* are:

Setting	**Description**
True	(Default) The axis is automatically scaled based on the data being charted.
False	The axis is not automatically scaled. Values in **DivisionsPerLabel** and **DivisionsPerTick** are used to determine the scale.

Visual Studio 6.0

Visual Basic: MSChart Control

Auto Property (Intersection)

See Also Example Applies To

Returns or sets a value that determines whether or not the **Intersection** object uses the value of the **Point** property to position the axis.

Syntax

object.**Auto** [= *boolean*]

The **Auto** 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 item is displayed, as described in Settings.

Settings

The settings for *boolean* are:

Setting	**Description**
True	(Default) The axis is positioned in its standard location.
False	The intersecting axis is positioned at the value indicated by **Point.**

Visual Studio 6.0

Visual Basic: MSChart Control

Auto Property (Label)

See Also Example Applies To

Returns or sets a value that determines whether axis labels are automatically rotated to improve the chart layout.

Syntax

object.**Auto** [= *boolean*]

The **Auto** 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 how axis labels will be positioned, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) The labels may be rotated.
False	The labels are not rotated. Long labels may not display properly.

Visual Studio 6.0

Visual Basic: MSChart Control

Auto Property (SeriesMarker)

See Also Example Applies To

Returns or sets a value that determines if the **SeriesMarker** object assigns the next available marker to all data points in the series.

Syntax

object.**Auto** [= *boolean*]

The **Auto** 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 how markers are assigned, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) The **SeriesMarker** object assigns the marker.
False	You can assign a custom marker.

Remarks

Set this property to **False** if you wish to change the series marker type.

This property is automatically set to **False** if the **Marker** property of the **DataPoint** object is set.

Visual Studio 6.0

Visual Basic: MSChart Control

Auto Property (ValueScale)

See Also Example Applies To

Returns or sets a value that determines whether automatic scaling is used to draw the value axis.

Syntax

object.**Auto** [= *boolean*]

The **Auto** 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 determines whether automatic scaling is used, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	The scale is automatically set based on the data being charted.
False	The values in the **Minimum**, **Maximum**, **MajorDivisions** and **MinorDivisions** properties are used to scale the axis.

Visual Basic Reference

Visual Studio 6.0

AutoActivate Property

See Also Example Applies To

Returns or sets a value that enables the user to activate an object by double-clicking the **OLE** container control or by moving the focus to the **OLE** container control.

Syntax

object.**AutoActivate** [= *value*]

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer or constant specifying the technique used to activate the object within the **OLE** container control, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
vbOLEActivateManual	0	Manual. The object isn't automatically activated. You can activate an object programmatically using the **DoVerb** method.
vbOLEActivateGetFocus	1	Focus. If the **OLE** container control contains an object that supports single click activation, the application that provides the object is activated when the **OLE** container control receives the focus.
vbOLEActivateDoubleclick	2	(Default) Double-Click. If the **OLE** container control contains an object, the application that provides the object is activated when the user double-clicks the **OLE** container control or presses ENTER when the control has the focus.
vbOLEActivateAuto	3	Automatic. If the **OLE** container control contains an object, the application that provides the object is activated based on the object's normal method of activation either when the control receives the focus or when the user double-clicks the control.

Remarks

You can determine if the **OLE** container control contains an object by checking the **OLEType** property.

Note When **AutoActivate** is set to 2 (Double-Click), the DblClick event doesn't occur when the user double-clicks an **OLE** container control.

Visual Basic: Windows Controls

Visual Studio 6.0

AutoBuddy Property

See Also Example Applies To

Sets or returns a value that determines whether the **UpDown** control automatically uses a control as its buddy control, based on its tab order.

Syntax

object.**AutoBuddy** [= *value*]

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

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
value	A boolean expression that determines the buddy control, as described in Settings.

Settings

The settings for *value* are:

Setting	**Description**
True	The **UpDown** control uses the previous control in the tab order as its buddy control. If no controls with a previous tab index can be used as a buddy control, the **UpDown** control uses the first available control with a higher tab index as its buddy control.
False	(Default) The **UpDown** control uses the setting in the **BuddyControl** property as its buddy control.

Remarks

Setting the **AutoBuddy** property to **True** also sets the **BuddyControl** property. Setting **AutoBuddy** to **False** clears the **BuddyControl** property.

Visual Basic: Multimedia MCI Control

Visual Studio 6.0

AutoEnable Property (Multimedia MCI Control)

See Also Example Applies To

Determines if the **Multimedia MCI** control can automatically enable or disable individual buttons in the control. If the **AutoEnable** property is set to **True**, the **Multimedia MCI** control enables those buttons that are appropriate for the current mode of the specified MCI device type. This property also disables those buttons that the current mode of the MCI device does not support.

Syntax

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

Remarks

The effect of the **AutoEnable** property is superseded by the **Enabled** property. The **AutoEnable** property can automatically enable or disable individual buttons in the control when the **Multimedia MCI** control is enabled (**Enabled** property set to **True**). When the **Enabled** property is **False**, keyboard and mouse run-time access to the **Multimedia MCI** control are turned off, regardless of the **AutoEnable** property setting.

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

Setting	Description
False	Does not enable or disable buttons. The program controls the states of the buttons by setting the **Enabled** and **ButtonEnabled** properties.
True	(Default) Enables buttons whose functions are available and disables buttons whose functions are not.

The following tables show how the MCI mode settings are reflected in the control's property settings.

Play mode

Record mode

Pause mode

Stop mode

Open mode

Seek or Not Ready modes

The effect of the **AutoEnable** property supersedes the effects of *Button***Enabled** properties. When the **Enabled** and **AutoEnable** properties are both **True**, the *Button***Enable** properties are not used.

Data Type

Integer (Boolean)

Visual Studio 6.0

Visual Basic: MSChart Control

AutoIncrement Property

See Also Example Applies To

Returns or sets a value that determines if the properties that set the current data point are incremented during data entry without manually setting the **Column** and **Row** properties.

Syntax

object.**AutoIncrement** [= *boolean*]

The **AutoIncrement** 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 current data point is incremented, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	When the **Data** property is changed, the **Row** property updates to the next row in the column. If you are at the end of a column, the **Column** property increments to the next column.
False	(Default) The current data point is not incremented.

Visual Studio 6.0

Visual Basic: MSChart Control

AutoLayout Property

See Also Example Applies To

Returns or sets a value that determines whether or not a **Plot** object is in manual or automatic layout mode.

Syntax

object.**AutoLayout** [= *boolean*]

The **AutoLayout** 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 the layout mode, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) The **Plot** object automatically determines the proper size and position of the plot based on the size and position of other elements.
False	The coordinates specified by **Plot** object's **LocationRect** property are used to position the plot.

Visual Studio 6.0

Visual Basic: MSChart Control

Automatic Property

See Also Example Applies To

Returns or sets a value that determines whether the color is calculated automatically. This is only used for edge pens on chart elements.

Syntax

object.**Automatic** [= *boolean*]

The **Automatic** 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 determines whether the color is calculated automatically, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	Color automatically picks up the brush color used on the chart series.
False	The color is determined based on the settings of **Value**.

Visual Basic: Windows Controls

Visual Studio 6.0

AutoPlay Property

See Also Example Applies To

Returns or sets a value which determines if the **Animation** control will begin to play an .avi file when the .avi file is loaded into the control.

Syntax

object.**Autoplay** [= *boolean*]

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

Part	Description
object	An object expression that evaluates to an **Animation** control.
boolean	A Boolean expression specifying whether the **Autoplay** Property is enabled.

Settings

The settings for *boolean* are:

Setting	Description
True	The .avi file plays automatically in a continuous loop once it is loaded into the **Animation** control.
False	An .avi file, once loaded, does not play until the **Play** method is used.

Data Type

Integer (Boolean)

Remarks

An .avi file played using the **Autoplay** property will continue to repeat until **Autoplay** is set to False.

Visual Basic Reference

Visual Studio 6.0

AutoRedraw Property

See Also Example Applies To

Returns or sets the output from a graphics method to a persistent graphic.

Syntax

object.**AutoRedraw** [= *boolean*]

The **AutoRedraw** 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 how the object is repainted, as described in Settings.

Settings

The settings for *boolean* are:

Setting	**Description**
True	Enables automatic repainting of a **Form** object or **PictureBox** control. Graphics and text are written to the screen and to an image stored in memory. The object doesn't receive Paint events; it's repainted when necessary, using the image stored in memory.
False	(Default) Disables automatic repainting of an object and writes graphics or text only to the screen. Visual Basic invokes the object's Paint event when necessary to repaint the object.

Remarks

This property is central to working with the following graphics methods: **Circle**, **Cls**, **Line**, **Point**, **Print**, and **PSet**. Setting **AutoRedraw** to **True** automatically redraws the output from these methods in a **Form** object or **PictureBox** control when, for example, the object is resized or redisplayed after being hidden by another object.

You can set **AutoRedraw** in code at run time to alternate between drawing persistent graphics (such as a background or grid) and temporary graphics. If you set **AutoRedraw** to **False**, previous output becomes part of the background screen.

When **AutoRedraw** is set to **False**, background graphics aren't deleted if you clear the drawing area with the **Cls** method. Setting **AutoRedraw** back to **True** and then using **Cls** clears the background graphics.

Note If you set the **BackColor** property, all graphics and text, including the persistent graphic, are erased. In general, all graphics should be displayed using the Paint event unless **AutoRedraw** is set to **True**.

To retrieve the persistent graphic created when **AutoRedraw** is set to **True**, use the **Image** property. To pass the persistent graphic to a Windows API when **AutoRedraw** is set to **True**, use the object's **hDC** property.

If you set a form's **AutoRedraw** property to **False** and then minimize the form, the **ScaleHeight** and **ScaleWidth** properties are set to icon size. When **AutoRedraw** is set to **True**, **ScaleHeight** and **ScaleWidth** remain the size of the restored window.

If **AutoRedraw** is set to **False**, the **Print** method will print on top of graphical controls such as the **Image** and **Shape** controls.

Visual Basic Reference

AutoRedraw Property Example

This example alternately displays two graphics on a **PictureBox** control: a persistent filled circle and temporary vertical lines. Click the **PictureBox** to draw or redraw the lines. Resizing the form requires the temporary graphic to be redrawn. To try this example, paste the code into the Declarations section of a form that has a **PictureBox** control named Picture1. Press F5 to run the program, and click the graphic each time you resize the form.

```
Private Sub Form_Load ()
   Picture1.ScaleHeight = 100   ' Set scale to 100.
   Picture1.ScaleWidth = 100
   Picture1.AutoRedraw = True    ' Turn on AutoRedraw.
   Picture1.ForeColor = 0   ' Set ForeColor.
   Picture1.FillColor = QBColor(9)   ' Set FillColor.
   Picture1.FillStyle = 0   ' Set FillStyle.
   Picture1.Circle (50, 50), 30   ' Draw a circle.
   Picture1.AutoRedraw = False   ' Turn off AutoRedraw.
End Sub

Private Sub Picture1_Click ()
   Dim I   ' Declare variable.
   Picture1.ForeColor = Rgb(Rnd * 255, 0, 0)   ' Select random color.
   For I = 5 To 95 Step 10   ' Draw lines.
      Picture1.Line (I, 0)-(I, 100)
   Next
End Sub
```

Visual Basic Reference

Visual Studio 6.0

AutoShowChildren Property

See Also Example Applies To

Returns or sets a value that determines whether MDI child forms are displayed when loaded.

Syntax

object.**AutoShowChildren** [= *boolean*]

The **AutoShowChildren** 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 MDI child forms are automatically visible, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) MDI child forms are automatically displayed when loaded.
False	MDI child forms aren't automatically displayed when loaded.

Remarks

You can use the **AutoShowChildren** property to load MDI child forms and leave them hidden until they're displayed using the **Show** method.

Visual Basic Reference

AutoShowChildren Property Example

This example presents an **MDIForm** object with an MDI child form, uses the **AutoShowChildren** property to create a hidden form as another instance of the MDI child form, and then creates a visible MDI child form. To try this example, set the **MDIChild** property to **True** on Form1, and then create an **MDIForm** with the Add MDI Form command on the Project menu. Copy the code into the Declarations section of the **MDIForm**, and then press F5 to run the program.

```
Private Sub MDIForm_Load()
   MDIForm1.AutoShowChildren = False   ' Set to hide child forms.
   Dim HideForm As New Form1   ' Declare new form.
   HideForm.Caption = "HideForm"   ' Set its caption.
   Load HideForm   ' Load it; it's hidden.
   MDIForm1.AutoShowChildren = True   ' Set to show child forms.
   Dim ShowForm As New Form1   ' Declare another new form.
   ShowForm.Caption = "ShowForm"   ' Set its caption.
   Load ShowForm   ' Load it; it's displayed.
End Sub
```

Visual Basic Reference

Visual Studio 6.0

AutoSize Property

See Also Example Applies To

Returns or sets a value that determines whether a control is automatically resized to display its entire contents.

Syntax

object.**AutoSize** [= *boolean*]

The **AutoSize** 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 control is resized, as described in Settings.

Settings

The settings for *boolean* are:

Setting	**Description**
True	Automatically resizes the control to display its entire contents.
False	(Default) Keeps the size of the control constant. Contents are clipped when they exceed the area of the control.

Visual Basic: Windows Controls

Visual Studio 6.0

AutoSize Property (Panel Object)

See Also Example Applies To

Returns or sets a value that determines the width of a **Panel** object after the **StatusBar** control has been resized.

Syntax

object.**AutoSize** [= *number*]

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

Part	Description
object	An object expression that evaluates to a **Panel** object.
number	A constant or value specifying the type of action, as described in Settings.

Settings

The settings for *number* are:

Constant	Value	Description
sbrNoAutoSize	0	(Default) None. No autosizing occurs. The width of the **Panel** is always and exactly that specified by the **Width** property.
sbrSpring	1	Spring. When the parent form resizes and there is extra space available, all panels with this setting divide the space and grow accordingly. However, the panels' width never falls below that specified by the **MinWidth** property.
sbrContents	2	Content. The **Panel** is resized to fit its contents, however, the width will never fall below the width specified by the **MinWidth** property.

Remarks

Panel objects with the Contents style have precedence over those with the Spring style. This means that a Spring-style **Panel** is shortened if a **Panel** with the Contents style requires that space.

Visual Basic: Windows Controls

AutoSize Property (Panel Object) Example

This example adds two **Panel** objects to a **StatusBar** control and sets the **AutoSize** property to Content for all panels. As the cursor is moved over the objects on the form, the x and y coordinates are displayed as well as the **Tag** property value for each control. To try the example, place a **StatusBar**, a **PictureBox**, and a **CommandButton** on a form, then paste the code into the Declarations section. Run the example and move the cursor over the various controls.

```
Private Sub Form_Load()
   Dim pnlX As Panel
   ' Set long tags for each object.
   Form1.Tag = "Project 1 Form"
   Command1.Tag = "A command button"
   Picture1.Tag = "Picture Box Caption"
   StatusBar1.Tag = "Application StatusBar1"
   ' Set the AutoSize style of the first panel to Contents.
   StatusBar1.Panels(1).AutoSize = sbrContents
   ' Add 2 more panels, and set them to Contents.
   Set pnlX = StatusBar1.Panels.Add
   pnlX.AutoSize = sbrContents
   Set pnlX = StatusBar1.Panels.Add
   pnlX.AutoSize = sbrContents
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
   ' Display the control's tag in panel 1, and x and y
   ' coordinates in panels 2 and 3. Because AutoSize = Contents,
   ' the first panel stretches to accommodate the varying text.
   StatusBar1.Panels(1).Text = Form1.Tag
   StatusBar1.Panels(2).Text = "X = " & x
   StatusBar1.Panels(3).Text = "Y = " & y
End Sub

Private Sub Command1_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
   StatusBar1.Panels(1).Text = Command1.Tag
   StatusBar1.Panels(2).Text = "X = " & x
   StatusBar1.Panels(3).Text = "Y = " & y
End Sub

Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
   StatusBar1.Panels(1).Text = Picture1.Tag
   StatusBar1.Panels(2).Text = "X = " & x
   StatusBar1.Panels(3).Text = "Y = " & y
End Sub

Private Sub StatusBar1_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
   StatusBar1.Panels(1).Text = StatusBar1.Tag
   StatusBar1.Panels(2).Text = "X = " & x
   StatusBar1.Panels(3).Text = "Y = " & y
End Sub
```

Visual Basic: MaskedEdit Control

Visual Studio 6.0

AutoTab Property

See Also Example Applies To

Determines whether or not the next control in the tab order receives the focus as soon as the **Text** property of the **Masked Edit** control is filled with valid data. The **Mask** property determines whether the values in the **Text** property are valid.

Syntax

[*form*.]**MaskedEdit**.**AutoTab**[= {**True** | **False**}]

Remarks

Automatic tabbing occurs only if all the characters defined by the **Mask** property are entered into the control, the characters are valid, and the **AutoTab** property is set to **True.**

Setting	Description
False	(Default) **AutoTab** is not on. A ValidationError event occurs when you enter more characters than are defined by the input mask.
True	**AutoTab** is on. When you enter all the characters defined by the input mask, focus goes to the next control in the tab sequence, and all subsequent characters entered are handled by the next control.

The **Masked Edit** control is considered filled when you enter the last valid character in the control, regardless of where the character is in the input mask. This property has no effect if the **Mask** property is set to the empty string ("").

Data Type

Integer (Boolean)

Visual Basic Reference

Visual Studio 6.0

AutoVerbMenu Property

See Also Example Applies To

Returns or sets a value that determines if a pop-up menu containing the object's verbs is displayed when the user clicks the **OLE** container control with the right mouse button.

Syntax

object.**AutoVerbMenu**[= *boolean*]

The **AutoVerbMenu** 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 whether a pop-up menu is displayed, as described in Settings.

Settings

The settings for *boolean* are:

Setting	**Description**
True	(Default) When the user clicks the **OLE** container control with the right mouse button, a pop-up menu is displayed, showing the commands the object supports.
False	No pop-up menu is displayed.

Remarks

When this property is set to **True**, Click events and MouseDown events don't occur when the **OLE** container control is clicked with the right mouse button.

In order to display your own menus, the **AutoVerbMenu** property must be set to **False**.

Visual Basic: RichTextBox Control

Visual Studio 6.0

AutoVerbMenu Property (RichTextBox Control)

See Also Example Applies To

Returns or sets a value that determines if a pop-up menu containing the selected objects verbs is displayed when the user clicks the OLE object with the right mouse button.

Syntax

object.**AutoVerbMenu** [= *value*]

The **AutoVerbMenu** property has the following parts:

Part	Description
object	An object expression that evaluates to a **RichTextBox** control.
value	A Boolean expression that specifies whether a menu is displayed, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	A pop-up menu is displayed when the user clicks the right mouse button on the object.
False	No pop-up menu is displayed.

Remarks

When this property is set to **True**, Click events and MouseDown events don't occur for the **RichTextBox** control when the OLE object is clicked with the right mouse button. Any other region of the control will generate the correct events.

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

AvailableSpace Property

See Also Example Applies To Specifics

Description

Returns the amount of space available to a user on the specified drive or network share.

Syntax

object.**AvailableSpace**

The *object* is always a **Drive** object.

Remarks

The value returned by the **AvailableSpace** property is typically the same as that returned by the **FreeSpace** property. Differences may occur between the two values for computer systems that support quotas.

The following code illustrates the use of the **AvailableSpace** property:

```
Sub ShowAvailableSpace(drvPath)
    Dim fs, d, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set d = fs.GetDrive(fs.GetDriveName(drvPath))
    s = "Drive " & UCase(drvPath) & " - "
    s = s & d.VolumeName  & vbCrLf
    s = s & "Available Space: " & FormatNumber(d.AvailableSpace/1024, 0)
    s = s & " Kbytes"
    MsgBox s
End Sub
```

Visual Studio 6.0

Visual Basic: MSChart Control

Axis Property

See Also Example Applies To

Returns a reference to an **Axis** object that describes an axis on a chart.

Syntax

object.**Axis**(*axisID*)

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

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
axisID	Required. An integer or VtChAxisId constant that identifies a specific axis.
index	Optional. Reserved for future use. Identifies the specific axis when there is more than one axis with the same *axisID*.

Remarks

Three axes are available: *x*, *y*, and *z*. The *z* axis is visible only when the chart is a 3D chart.

Visual Studio 6.0

Visual Basic: MSChart Control

AxisGrid Property

See Also Example Applies To

Returns a reference to an **AxisGrid** object that represents the planar area surrounding a chart axis.

Syntax

object.**AxisGrid**

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

AxisId Property

See Also Example Applies To

Returns a specific axis that intersects with the current axis.

Syntax

object.**AxisId**

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

Return Value

The return value is an integer that identifies the intersecting axis.

Visual Studio 6.0

Visual Basic: MSChart Control

AxisScale Property

See Also Example Applies To

Returns a reference to an **AxisScale** object that describes how chart values are plotted on an axis.

Syntax

object.**AxisScale**

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

AxisTitle Property

See Also Example Applies To

Returns a reference to an **AxisTitle** object associated with the axis of a chart.

Syntax

object.**AxisTitle**

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