Visual Basic: Winsock Control

Visual Studio 6.0

DataArrival Event

See Also Example Applies To

Occurs when new data arrives.

Syntax

*object*_**DataArrival** (*bytesTotal* **As Long**)

The DataArrival event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
bytesTotal	Long. The total amount of data that can be retrieved.

Remarks

This event will not occur if you do not retrieve all the data in one **GetData** call. It is activated only when there is new data. Use the **BytesReceived** property to check how much data is available at any time.

Visual Basic: Winsock Control

GetData Method (WinSock Control), DataArrival Event Example

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

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

Visual Basic: RDO Data Control

Visual Studio 6.0

DataChanged Event

See Also Example Applies To

Occurs when the value of the column has changed.

Syntax

Private Sub *object*.**DataChanged**( )

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

Remarks

This event is raised after the data in a column has been changed. The new data can be accessed through the **rdoColumn** object's **Value** property. You can also use the WillChange event to prevent or modify the change about to be made on a column-by-column basis. However, once the DataChanged event fires, the change has already been committed to the database.

Visual Basic: DataRepeater Control

Visual Studio 6.0

DataUpdate Event

See Also Example Applies To

Occurs when a field is edited either programmatically or by a user.

Syntax

*Private Sub object*_**DataUpdate(***cancel* **As boolean**, **ByVal** *Record* **As Variant**, **ByVal** *field* **As String)**

The DataUpdate event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
cancel	A Boolean value that specifies if the operation is canceled.
record	A Variant array that contains the bookmark of the current record.
field	A string that indicates the name of the field which is about to change.

Remarks

The *field* argument does not have to be a bound field. If any field in the recordset is changed, the event will occur.

Visual Studio 6.0

Visual Basic: MSChart Control

DataUpdated Event

See Also Example Applies To

Occurs when the chart data grid has changed.

Syntax

Private Sub *object*_**DataUpdated** (*row* **As Integer**, *column* **As Integer**, *labelRow* **As Integer**, *labelColumn* **As Integer**, *labelSetIndex* **As Integer**, *updateFlags* **As Integer**)

The DataUpdated event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
row	Integer. Indicates the row in the data grid.
column	Integer. Indicates the column in the datagrid.
labelRow	Integer. Indicates the row label.
labelColumn	Integer. Indicates the column label.
labelSetIndex	Integer. Identifies the level of labels. Levels of labels are numbered from the axis out, beginning with 1.
updateFlags	Integer. Provides information about the update of the data, as described in Settings.

Settings

The following table lists the constants for *updateFlags*.

Constant	Description
VtChNoDisplay	Absence of update flags; the chart display is not affected. (Defined as 0.)
VtChDisplayPlot	Update will cause the plot to repaint.
VtChLayoutPlot	Update will cause the plot to lay out.

VtChDisplayLegend	Update will cause the legend to repaint.
VtChLayoutLegend	Update will cause the legend to lay out.
VtChLayoutSeries	Update will cause the series to lay out.
VtChPositionSection	A chart section has been moved or resized.

Remarks

If row and column are nonzero, the change occurs to the indicated data cell. If *labelRow* or *labelColumn*, along with *labelSetIndex*, are nonzero, the indicated row or column label changes. If none of these are nonzero, no specific information about the change is available.

Visual Basic: Windows Controls

Visual Studio 6.0

DateClick Event

See Also Example Applies To

Occurs when a date on the control is clicked.

Syntax

Private Sub *object*_**DateClick([***index* **As Integer],** *DateClicked* **As Date)**

The DateClick event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
DateClicked	A date expression specifying the date that was clicked.

Remarks

The DateClick event can be used to respond to the user clicking on a particular date. The *DateClicked* can be used to determine which date was clicked.

Visual Basic: Windows Controls

Visual Studio 6.0

DateDblClick Event

See Also Example Applies To

Occurs when a date on the control is double-clicked.

Syntax

Private Sub *object*_**DateDblClick([***index* **As Integer],** *DateDblClicked* **As Date)**

The DateDblClick event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
DateDblClicked	A date expression specifying the date that was clicked.

Remarks

The DateDblClick event can be used to respond to the user double-clicking on a particular date. The *DateDblClicked* can be used to determine which date was double-clicked.

Visual Basic Reference

Visual Studio 6.0

DblClick Event (ActiveX Controls)

See Also Example Applies To

Occurs when the user presses and releases a mouse button and then presses and releases it again over an object.

Syntax

Private Sub *object*_**DblClick (***index* **As Integer)**

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Identifies the control if it's in a control array.

Remarks

The argument *Index* uniquely identifies a control if it's in a control array. You can use a DblClick event procedure for an implied action, such as double-clicking an icon to open a window or document. You can also use this type of procedure to carry out multiple steps with a single action, such as double-clicking to select an item in a list box and to close the dialog box.

To produce such shortcut effects in Visual Basic, you can use a DblClick event procedure for a list box or file list box in tandem with a default button a **CommandButton** control with its **Default** property set to **True**. As part of the DblClick event procedure for the list box, you simply call the default button's Click event.

For those objects that receive Mouse events, the events occur in this order: MouseDown, MouseUp, Click, DblClick, and MouseUp.

If DblClick doesn't occur within the system's double-click time limit, the object recognizes another Click event. The double-click time limit may vary because the user can set the double-click speed in the Control Panel. When you're attaching procedures for these related events, be sure that their actions don't conflict. Controls that don't receive DblClick events may receive two clicks instead of a DblClick.

Note To distinguish between the left, right, and middle mouse buttons, use the MouseDown and MouseUp events.

If there is code in the Click event, the DlbClick event will never trigger.

Visual Basic Reference

DblClick Event (ActiveX Controls) Example

This example displays a selected list item in a **TextBox** control when either a **CommandButton** control is clicked or a list item is double-clicked. To try this example, paste the code into the Declarations section of a **Form** object that contains a **ListBox** control, a **TextBox** control, and a **CommandButton** control. Then run the example and click the **CommandButton** control or double-click an item in the **ListBox** control.

```
Private Sub Form_Load ()
   List1.AddItem "John"   ' Add list box entries.
   List1.AddItem "Paul"
   List1.AddItem "George"
   List1.AddItem "Ringo"
End Sub

Private Sub List1_DblClick ()
   Command1.Value = True   ' Trigger Click event.
End Sub

Private Sub Command1_Click ()
   Text1.Text = List1.Text   ' Display selection.
End Sub
```

Visual Basic: DataCombo/DataList Controls

Visual Studio 6.0

DblClick Event (DBCombo Control)

See Also Example Applies To

Occurs when the user double-clicks the **DBCombo** control with the mouse button.

Syntax

Private Sub *object*_**DblClick ([***index* **As Integer,]** *Area* **As Integer)**

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
Area	An integer expression that specifies where the control was double clicked, as described below.

The *Area* parameter can contain the following values:

Constant	Value	Description
dbcAreaButton	0	The user double-clicked the button on the **DBCombo** control.
dbcAreaEdit	1	The user double-clicked in the text box part of the **DBCombo** control.
dbcAreaList	2	The user double-clicked in the list part of the **DBCombo** control. (This only occurs when the **Style** property is set to 1)

Remarks

You can use a DblClick event procedure for an implied action or use it to carry out multiple steps with a single action.

If DblClick doesn't occur within the system's double-click time limit, the object recognizes another Click event. The double-click time limit may vary because the user can set the double-click speed in the Control Panel.

Note To distinguish between the left, right, and middle mouse buttons, use the MouseDown and MouseUp events.

Visual Basic Reference

Visual Studio 6.0

DblClick Event

See Also Example Applies To

Occurs when the user presses and releases a mouse button and then presses and releases it again over an object.

For a form, the DblClick event occurs when the user double-clicks a disabled control or a blank area of a form. For a control, it occurs when the user:

- Double-clicks a control with the left mouse button.
- Double-clicks an item in a **ComboBox** control whose **Style** property is set to 1 (Simple) or in a **FileListBox**, **ListBox**, **DataCombo**, or **DataList** control.

Syntax

Private Sub Form_DblClick ( )

Private Sub *object*_**DblClick (***index* **As Integer)**

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Identifies the control if it's in a control array.

Remarks

The argument *Index* uniquely identifies a control if it's in a control array. You can use a DblClick event procedure for an implied action, such as double-clicking an icon to open a window or document. You can also use this type of procedure to carry out multiple steps with a single action, such as double-clicking to select an item in a list box and to close the dialog box.

To produce such shortcut effects in Visual Basic, you can use a DblClick event procedure for a list box or file list box in tandem with a default button a **CommandButton** control with its **Default** property set to **True**. As part of the DblClick event procedure for the list box, you simply call the default button's Click event.

For those objects that receive Mouse events, the events occur in this order: MouseDown, MouseUp, Click, DblClick, and MouseUp.

If DblClick doesn't occur within the system's double-click time limit, the object recognizes another Click event. The double-click time limit may vary because the user can set the double-click speed in the Control Panel. When you're attaching procedures for these related events, be sure that their actions don't conflict. Controls that don't receive DblClick events may receive two clicks instead of a DblClick.

When debugging events, do not use MsgBox statements to show when the event occurred, as this will disturb the normal functioning of many events. (For example, a MsgBox in the Click event will prevent DblClick from being raised.) Instead, use Debug.Print to show the order in which events occur

Note To distinguish between the left, right, and middle mouse buttons, use the MouseDown and MouseUp events.

Visual Basic Reference

DblClick Event Example

This example displays a selected list item in a **TextBox** control when either a **CommandButton** control is clicked or a list item is double-clicked. To try this example, paste the code into the Declarations section of a **Form** object that contains a **ListBox** control, a **TextBox** control, and a **CommandButton** control. Then run the example and click the **CommandButton** control or double-click an item in the **ListBox** control.

```
Private Sub Form_Load ()
   List1.AddItem "John"   ' Add list box entries.
   List1.AddItem "Paul"
   List1.AddItem "George"
   List1.AddItem "Ringo"
End Sub

Private Sub List1_DblClick ()
   Command1.Value = True   ' Trigger Click event.
End Sub

Private Sub Command1_Click ()
   Text1.Text = List1.Text   ' Display selection.
End Sub
```

Visual Basic Reference

Visual Studio 6.0

DECommandAdded Event

See Also Example Applies To

Occurs after a **DECommand** object is added to a DataEnvironment object.

Syntax

Sub *object*_**DECommandAdded(***value***)**

The **DECommandAdded** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
value	A value that specifies the DECommand object that was added to the DataEnvironment object.

Visual Basic Reference

Visual Studio 6.0

DECommandPropertyChanged Event

See Also Example Applies To

Occurs after any property of the DECommand object is changed.

Syntax

Sub *object*_**DECommandPropertyChanged(***value, string***)**

The **DECommandPropertyChanged** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
value	A value that specifies the DECommand object that had its property value changed.
string	A string expression that specifies the name of the property that was changed.

Remarks

If multiple properties are changed at once, this event generates once for each modified property.

Visual Basic Reference

Visual Studio 6.0

DECommandRemoved Event

See Also Example Applies To

Occurs after a **DECommand** object is removed from the DataEnvironment object.

Syntax

*Object*_**DECommandRemoved(***value***)**

The **DECommandRemoved** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
value	A value that specifies the **DECommand** object that was removed.

Visual Basic Reference

Visual Studio 6.0

DEConnectionAdded Event

See Also Example Applies To

Occurs after a DEConnection object is added to the DataEnvironment object.

Syntax

Sub *object*_**DEConnectionAdded(***value***)**

The **DEConnectionAdded** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
value	A value that specifies the DEConnection object that was added to the DataEnvironment object.

Remarks

This event occurs each time a **DEConnection** object is added to the DataEnvironment object, regardless of whether it was added through the user interface or using the **Add** method through the Extensibility Object Model.

Visual Basic Reference

Visual Studio 6.0

DEConnectionPropertyChanged Event

See Also Example Applies To

Occurs after any property of the **DEConnection** object is changed.

Syntax

Sub *object*_**DEConnectionPropertyChanged(***value, string***)**

The DEConnectionPropertyChanged event syntax has these parts:

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
value	A value that specifies the **DEConnection** object that had its property value changed.
string	A string expression that specifies the name of the property that was changed.

Remarks

If multiple properties are changed at once, this event generates for each modified property.

Visual Basic Reference

Visual Studio 6.0

DEConnectionRemoved Event

See Also Example Applies To

Occurs after a DEConnection object is removed from the DataEnvironment object.

Syntax

Sub *object*.**ConnectionRemoved(***value***)**

The ConnectionRemoved event syntax has these parts:

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
value	A value that specifies the removed DEConnection object.

Visual Basic: SysInfo Control

Visual Studio 6.0

DeviceArrival Event

See Also Example Applies To

Occurs when a new device is added to the system.

Syntax

Private Sub *object*_**DeviceArrival(**[*index* **As Integer,**] **ByVal** *devicetype* **As Long**, **ByVal** *deviceid* **As Long**, **ByVal** *devicename* **As String**, **ByVal** *devicedata* **As Long)**

The DeviceArrival event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	A numeric expression that evaluates to the index of a control if it is in a control array.
devicetype	A value that indicates the type of device, as described in Settings.
deviceid	Returns a value that identifies the device. The returned value depends on the value of *devicetype*. See Settings below.
devicename	Returns a string or **Null** depending on the value of *devicetype*. If *devicetype* returns 0-2 or 4, then *devicename* returns **Null**. When *devicetype* is 3, then *devicename* returns the port's friendly name. For example, "COM1", "Hayes 2400 Smartmodem", and "LPT1" are considered friendly names.
devicedata	A value as described in Settings. The value returned depends on the value of *devicetype*.

Settings

The possible values for *devicetype* are:

Value	Description
0	OEM-defined device type.
1	Device node (Windows 95 or later). A device node refers to a device that can host other hardware, such as a SCSCI controller.

2	Logical volume (disk drive).
3	Serial or parallel port.
4	Unsupported.

The settings for *deviceid* depend on the setting of *devicetype*:

If *devicetype* is:	**Then *deviceid* returns**
0	Globally unique identifier (GUID) for the device.
1	Device node number.
2	Logical unit mask identifying one or more logical units. Each bit in the mask corresponds to one logical drive. Bit 0 represents drive A, bit 1 drive B, and so on. For example, 1 = Drive A, 8 = Drive D, and 128 = Drive H.
3	**Null**
4	Unsupported.

The settings for *devicedata* depend on the value of *devicetype*:

If *devicetype* is:	**Then *devicedata* returns**
0	OEM-specific function value. Possible values depend on the device.
1	**Null**
2	Either 1 or 2, depending on the type of logical drive. 1. Media. For example, a CD-ROM. 2Net. Indicated logical volume is a network volume.
3	**Null**
4	Unknown.

Remarks

This event is very useful if your application can dynamically make use of new hardware.

Visual Basic: SysInfo Control

Visual Studio 6.0

DeviceOtherEvent Event

See Also Example Applies To

A notification event that does not map onto the general events.

Syntax

Private Sub *object*_**DeviceOtherEvent(**[*index* **As Integer,**] **ByVal** *devicetype* **As Long**, **ByVal** *eventname* **As String**, **ByVal** *datapointer* **As Long)**

The DeviceOtherEvent event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
Index	A numeric expression that evaluates to the index of a control if it is in a control array.
Devicetype	A value that indicates the type of device that will process events, as described in Settings.
Eventname	A string expression that evaluates to the name of an event.
Datapointer	A long value that points to device-specific data.

Settings

The settings for *devicetype* are:

Value	Description
0	OEM-defined device type.
1	Devnode number (Windows 95 or later).
2	Logical volume (disk drive).
3	Serial or parallel port.
4	Unsupported.

Visual Basic: SysInfo Control

Visual Studio 6.0

DeviceQueryRemove Event

See Also Example Applies To

Occurs just before a device is removed from the system.

Syntax

Private Sub *object*_**DeviceQueryRemove(**[*index* **As Integer,**] **ByVal** *devicetype* **As Long**, **ByVal** *deviceid* **As Long**, **ByVal** *devicename* **As String**, **ByVal** *devicedata* **As Long,** *cancel* **As Boolean)**

The DeviceQueryRemove event syntax has these parts:

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Index	A numeric expression that evaluates to the index of a control if it is in a control array.
Devicetype	A value that indicates the type of device that has been added, as described in Settings.
Deviceid	Returns a value that identifies the device. The returned value depends on the value of *devicetype*. See Settings below.
Devicename	**Null** for all settings of *devicetype* except **DeviceTypePort**. When *devicetype* is **DeviceTypePort**, then *devicename* is dbcp_name.
Devicedata	Returns a string or **Null** depending on the value of *devicetype*. If *devicetype* returns 0-2 or 4, then *devicename* returns **Null**. When *devicetype* is 3, then *devicename* returns the port's friendly name. For example, "COM1", "Hayes 2400 Smartmodem", and "LPT1" are considered friendly names.
Cancel	A Boolean value, as described in Settings.

Settings

The settings for *devicetype* are:

Value	Description
0	OEM-defined device type.

1	Devnode number (Windows 95 or later).
2	Logical volume (disk drive).
3	Serial or parallel port.
4	Unsupported.

The settings for *deviceid* are:

If *devicetype* is:	**Then *deviceid* returns**
0	Globally unique identifier (GUID) for the device.
1	Device node number.
2	Logical unit mask identifying one or more logical units. Each bit in the mask corresponds to one logical drive. Bit 0 represents drive A, bit 1 drive B, and so on. For example, 1 = Drive A, 8 = Drive D, and 128 = Drive H.
3	**Null**
4	Unsupported.

The settings for *devicedata* are:

If *devicetype* is:	**Then *devicedata* returns**
0	OEM-specific function value. Possible values depend on the device.
1	**Null**
2	Either 1 or 2, depending on the type of logical drive. 1. Media. For example, a CD-ROM. 2Net. Indicated logical volume is a network volume.
3	**Null**
4	Unknown.

The settings for *cancel* are:

Setting	**Description**

True	The system is preventing the device from being removed.
False	The system allows the device to be removed.

Remarks

This event takes place only if the process responsible for removing the device sends the required message to the operating system before the device is removed.

Visual Basic: SysInfo Control

Visual Studio 6.0

DeviceQueryRemoveFailed Event

See Also Example Applies To

Occurs if code in the DeviceQueryRemove event cancelled the removal of a device.

Syntax

Private Sub *object*_**DeviceQueryRemoveFailed(**[*index* **As Integer,**] **ByVal** *devicetype* **As Long**, **ByVal** *deviceid* **As Long**, **ByVal** *devicename* **As String**, **ByVal** *devicedata* **As Long,** *cancel* **As Boolean)**

The DeviceQueryRemoveFailed event syntax has these parts:

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Index	A numeric expression that evaluates to the index of a control if it is in a control array.
Devicetype	Returns a value that identifies the device. The returned value depends on the value of *devicetype*. See Settings below.
Deviceid	A value that identifies the device, as described in Settings.
Devicename	Returns a string or **Null** depending on the value of *devicetype*. If *devicetype* returns 0-2 or 4, then *devicename* returns **Null**. When *devicetype* is 3, then *devicename* returns the port's friendly name. For example, "COM1", "Hayes 2400 Smartmodem", and "LPT1" are considered friendly names.
Devicedata	A value as described in Settings. The value returned depends on the value of *devicetype*.
Cancel	A Boolean value, as described in Settings.

Settings

The settings for *devicetype* are:

Value	Description
0	OEM-defined device type.
1	Device node (Windows 95 or later). A device node refers to a device that can host other hardware, such as a

	SCSCI controller.
2	Logical volume (disk drive).
3	Serial or parallel port.
4	Unsupported.

The settings for *deviceid* depend on the setting of *devicetype*:

If *devicetype* is:	**Then *devicedata* returns**
0	Globally unique identifier (GUID) for the device.
1	Device node number.
2	Logical unit mask identifying one or more logical units. Each bit in the mask corresponds to one logical drive. Bit 0 represents drive A, bit 1 drive B, and so on. For example, 1 = Drive A, 8 = Drive D, and 128 = Drive H.
3	**Null**
4	Unsupported.

The settings for *devicedata* depend on the value of *devicetype*:

If *devicetype* is:	**Then *devicedata* returns**
0	OEM-specific function value. Possible values depend on the device.
1	**Null**
2	Either 1 or 2, depending on the type of logical drive. 1. Media. For example, a CD-ROM. 2Net. Indicated logical volume is a network volume.
3	**Null**
4	Unknown.

The settings for *cancel* are:

Setting	**Description**

True	The system is preventing the device from being removed.
False	The system allows the device to be removed.

Visual Basic: SysInfo Control

Visual Studio 6.0

DeviceRemoveComplete Event

See Also Example Applies To

Occurs after a device is removed.

Syntax

Private Sub *object*_**DeviceQueryRemove(**[*index* **As Integer,**] **ByVal** *devicetype* **As Long**, **ByVal** *deviceid* **As Long**, **ByVal** *devicename* **As String**, **ByVal** *devicedata* **As Long,** *cancel* **As Boolean)**

The DeviceRemoveComplete event syntax has these parts:

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Index	A numeric expression that evaluates to the index of a control if it is in a control array.
Devicetype	A value that indicates the type of device that has been added, as described in Settings.
Deviceid	Returns a value that identifies the device. The returned value depends on the value of *devicetype*. See Settings below.
Devicename	Returns a string or **Null** depending on the value of *devicetype*. If *devicetype* returns 0-2 or 4, then *devicename* returns **Null**. When *devicetype* is 3, then *devicename* returns the port's friendly name. For example, "COM1", "Hayes 2400 Smartmodem", and "LPT1" are considered friendly names.
Devicedata	A value as described in Settings. The value returned depends on the value of *devicetype*.
Cancel	A Boolean value, as described in Settings.

Settings

The settings for *devicetype* are:

Value	Description
0	OEM-defined device type.
1	Device node (Windows 95 or later). A device node refers to a device that can host other hardware, such as a

	SCSCI controller.
2	Logical volume (disk drive).
3	Serial or parallel port.
4	Unsupported.

The settings for *deviceid* depend on the setting of *devicetype*:

If *devicetype* is:	**Then *deviceid* returns**
0	Globally unique identifier (GUID) for the device.
1	Device node number.
2	Logical unit mask identifying one or more logical units. Each bit in the mask corresponds to one logical drive. Bit 0 represents drive A, bit 1 drive B, and so on. For example, 1 = Drive A, 8 = Drive D, and 128 = Drive H.
3	**Null**
4	Unsupported.

The settings for *devicedata* depend on the value of *devicetype*:

If *devicetype* is:	**Then *devicedata* returns**
0	OEM-specific function value. Possible values depend on the device.
1	**Null**
2	Either 1 or 2, depending on the type of logical drive. 1. Media. For example, a CD-ROM. 2Net. Indicated logical volume is a network volume.
3	**Null**
4	Unknown.

The settings for *cancel* are:

Setting	**Description**

True	The system is preventing the device from being removed.
False	The system allows the device to be removed.

Remarks

In some cases this event occurs even though no other device removal event occurred.

Visual Basic: SysInfo Control

Visual Studio 6.0

DeviceRemovePending Event

See Also Example Applies To

Occurs after all applications have given approval to remove a device and the device is about to be removed.

Syntax

Private Sub *object*_**DeviceRemovePending(**[*index* **As Integer,**] **ByVal** *devicetype* **As Long**, **ByVal** *deviceid* **As Long**, **ByVal** *devicename* **As String**, **ByVal** *devicedata* **As Long,** *cancel* **As Boolean)**

The DeviceRemovePending event syntax has these parts:

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Index	A numeric expression that evaluates to the index of a control if it is in a control array.
Devicetype	A value that indicates the type of device that has been added, as described in Settings.
Deviceid	Returns a value that identifies the device. The returned value depends on the value of *devicetype*. See Settings below.
Devicename	Returns a string or **Null** depending on the value of *devicetype*. If *devicetype* returns 0-2 or 4, then *devicename* returns **Null**. When *devicetype* is 3, then *devicename* returns the port's friendly name. For example, "COM1", "Hayes 2400 Smartmodem", and "LPT1" are considered friendly names.
Devicedata	A value as described in Settings. The value returned depends on the value of *devicetype*.
Cancel	A Boolean value, as described in Settings.

Settings

The settings for *devicetype* are:

Value	Description
0	OEM-defined device type.
1	Device node (Windows 95 or later). A device node refers to a device that can host other hardware, such as a

	SCSCI controller.
2	Logical volume (disk drive).
3	Serial or parallel port.
4	Unsupported.

The settings for *deviceid* depend on the setting of *devicetype*:

If *devicetype* is:	**Then *deviceid* returns**
0	Globally unique identifier (GUID) for the device.
1	Device node number.
2	Logical unit mask identifying one or more logical units. Each bit in the mask corresponds to one logical drive. Bit 0 represents drive A, bit 1 drive B, and so on. For example, 1 = Drive A, 8 = Drive D, and 128 = Drive H.
3	**Null**
4	Unsupported.

The settings for *devicedata* depend on the value of *devicetype*:

If *devicetype* is:	**Then *devicedata* returns**
0	OEM-specific function value. Possible values depend on the device.
1	**Null**
2	Either 1 or 2, depending on the type of logical drive. 2. Media. For example, a CD-ROM. 2Net. Indicated logical volume is a network volume.
3	**Null**
4	Unknown.

The settings for *cancel* are:

Setting	**Description**

True	The system is preventing the device from being removed.
False	The system allows the device to be removed.

Visual Basic: SysInfo Control

Visual Studio 6.0

DevModeChange Event

See Also Example Applies To

Occurs when the user changes device mode settings.

Syntax

Private Sub *object*_**DevModeChange(**[*index* **As Integer,**] **ByVal** *devicename* **As String)**

The DevModeChange event syntax has these parts:

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Index	A numeric expression that evaluates to the index of a control if it is in a control array.
Devicename	A string expression that identifies a device name specified in the Windows registry.

Visual Basic: RDO Data Control

Visual Studio 6.0

Disconnect Event

See Also Example Applies To

Occurs after a connection has been closed.

Private Sub *object*.**Disconnect**( )

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

Remarks

Fired after a physical connection is closed. The developer can catch this event to do any clean-up work necessary.

Applies to **rdoConnection** object.

Visual Basic: SysInfo Control

Visual Studio 6.0

DisplayChanged Event

See Also Example Applies To

Occurs when system screen resolution changes.

Syntax

Private Sub *object*_**DisplayChanged(**[*index* **As Integer**]**)**

The DisplayChanged event syntax has these parts:

Part	Description
Object	An object expression that evaluates to an object in the Applies To list.
Index	A numeric expression that evaluates to the index of a control if it is in a control array.

Remarks

Use this event to make any adjustments you need to the interface of your application due to changes in screen resolution.

Visual Basic: SysInfo Control

DisplayChanged Event Example

This example tests the size of the active form after a change in screen resolution, and adjusts the size of the form if it exceeds the visible screen area. To run this example, put a **SysInfo** control on a form. Paste this code into the DisplayChanged event of the **SysInfo** control. Run the example then change the screen resolution.

```
Private Sub SysInfo1_DisplayChanged()
    If Screen.ActiveForm.Width > SysInfo1.WorkAreaWidth Then
        Screen.ActiveForm.Left = SysInfo1.WorkAreaLeft
        Screen.ActiveForm.Width = SysInfo1.WorkAreaWidth
    End If
    If Screen.ActiveForm.Height > SysInfo1.WorkAreaHeight Then
        Screen.ActiveForm.Top = SysInfo1.WorkAreaTop
        Screen.ActiveForm.Height = SysInfo1.WorkAreaHeight
    End If
End Sub
```

Visual Basic: RDO Data Control

Visual Studio 6.0

Dissociate Event

See Also Example Applies To

Occurs after an **rdoResultset** object has been dissociated from a connection.

Private Sub *object*.**Dissociate**( )

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

This event is raised after the **ActiveConnection** property has been set to **Nothing** and the result set has been dissociated from its connection.

Visual Basic Extensibility Reference

Visual Studio 6.0

DoGetNewFileName Event

See Also Example Applies To

Occurs whenever a Save As operation is performed on any component or project, whether manually performed from the **File** menu, or programmatically performed.

Syntax

Sub DoGetNewFileName(*vbproject* **As VBProject**, *filetype* **As vbext_FileType**, *newname* **As String**, *oldname* **As String**, *canceldefault* **As Boolean)**

The DoGetNewFileName event syntax has these parts:

Part	Description
vbproject	A **VBProject** object specifying the name of the project which will be written.
filetype	An enumerated value (**vbext_FileType**) specifying the type of file to be written, as listed in Settings.
newname	A string expression specifying the name of the new file. The file specification must be relative to the current **LastUsedPath** property or a fully qualified filename.
oldname	A string expression specifying the old name of the file.
canceldefault	A Boolean expression that determines the default Visual Basic action, as described in Settings.

Settings

The enumerated values for **vbext_FileType** are:

Constant	Value	Description
vbext_ft_Form	0	File type is a form.
vbext_ft_Module	1	File type is a basic module.
vbext_ft_Class	2	File type is a class module.
vbext_ft_Project	3	File type is a project.

vbext_ft_Exe	4	File type is an executable file.
vbext_ft_Res	6	File type is a resource file.
vbext_ft_UserControl	7	File type is a **User** control.
vbext_ft_PropertyPage	8	File type is a **Property Page**.
vbext_ft_DocObject	9	File type is a **User Document**.
vbext_ft_Binary	10	File type is a binary file.
vbext_ft_GroupProject	11	File type is a group project.
vbext_ft_Designer	12	File type is a designer object.

The settings for *canceldefault* are:

Setting	Description
True	Stops triggering this event for any subsequent add-ins connected to the **FileControl** object. If *newname* is a zero-length string ("") when *canceldefault* is set to **True**, the event is canceled; otherwise, the name entered in *newname* is used as the new filename.
False	Continues triggering this event for subsequent add-ins connected to the **FileControl** object. If no add-in sets *canceldefault* to **True**, the **Save File As** or **Make .Exe** dialog box is displayed with the string you entered in *newname* selected.

Remarks

If the *canceldefault* parameter is set to **True**, the **Save File As** dialog box is not displayed. If *canceldefault* is set to **False**, the **Save File As** dialog box displays. If more than one add-ins is connected, and *canceldefault* is set to **True** at any time during a Save As operation, the S**ave File As** dialog box will not display for any of the add-ins until the next Save As operation is performed.

The *newname* argument is initially set to the same value as *oldname*, but any add-in that receives this event can change it. One way to do this is through a custom user interface where you obtain the new name of the file and set *newname* to the user's selection. However, if *canceldefault* is **True** (meaning that a previous add-in has set it to **True**), you shouldn't set *newname* again.

This event occurs in all add-ins that are connected to the **FileControl** object. The add-in cannot prevent the file from being written to disk because the operation is complete. However, you can use this event to perform other tasks, such as:

- Log information about the event.
- Update information about the file.
- Back up the file.

Visual Basic: Multimedia MCI Control

Visual Studio 6.0

Done Event (Multimedia MCI Control)

See Also Example Applies To

Occurs when an **MCI** command for which the **Notify** property is **True** finishes.

Syntax

Private Sub *MMControl*_**Done** (*NotifyCode* **As Integer**)

Remarks

The *NotifyCode* argument indicates whether the **MCI** command succeeded. It can take any of the following settings.

Value	Setting/Result
1	**mciSuccessful** Command completed successfully.
2	**mciSuperseded** Command was superseded by another command.
4	**mciAborted** Command was aborted by the user.
8	**mciFailure** Command failed.

Visual Studio 6.0

Visual Basic: MSChart Control

DonePainting Event

See Also Example Applies To

Occurs immediately after the chart repaints or redraws.

Syntax

Private Sub *object*_**DonePainting** ()

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

Visual Basic: Windows Controls

Visual Studio 6.0

DownClick Event

See Also Example Applies To

This event occurs when the down or left arrow button is clicked.

Syntax

Private Sub *object*_**DownClick(**[*index* as integer]**)**

The DownClick event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

The DownClick event occurs after the Change event.

Visual Basic Reference

Visual Studio 6.0

DragDrop Event

See Also Example Applies To

Occurs when a drag-and-drop operation is completed as a result of dragging a control over an object and releasing the mouse button or using the **Drag** method with its *action* argument set to 2 (Drop).

Syntax

```
Private Sub Form_DragDrop(source As Control, x As Single, y As Single)

Private Sub MDIForm_DragDrop(source As Control, x As Single, y As Single)

Private Sub object_DragDrop([index As Integer,]source As Control, x As Single, y As Single)
```

The DragDrop event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
source	The control being dragged. You can include properties and methods in the event procedure with this argument for example, `Source.Visible = 0`.
x, *y*	A number that specifies the current horizontal (*x*) and vertical (*y*) position of the mouse pointer within the target form or control. These coordinates are always expressed in terms of the target's coordinate system as set by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties.

Remarks

Use a DragDrop event procedure to control what happens after a drag operation is completed. For example, you can move the source control to a new location or copy a file from one location to another.

When multiple controls can potentially be used in a *source* argument:

- Use the **TypeOf** keyword with the **If** statement to determine the type of control used with *source*.
- Use the control's **Tag** property to identify a control, and then use a DragDrop event procedure.

Note Use the **DragMode** property and **Drag** method to specify the way dragging is initiated. Once dragging has been initiated, you can handle events that precede a DragDrop event with a DragOver event procedure.

Visual Basic Reference

DragDrop Event Example

This example demonstrates the visual effect of dropping a **PictureBox** control onto another **PictureBox** control. To try this example, paste the code into the Declarations section of a form that contains three **PictureBox** controls. Set the **DragMode** property for Picture1 and Picture2 to 1 (Automatic). Use the **Picture** property to assign bitmaps to Picture1 and Picture2, and then press F5 and drag Picture1 or Picture2 over Picture3.

```
Private Sub Picture3_DragDrop (Source As Control, X as Single, Y As Single)
   If TypeOf Source Is PictureBox Then
      ' Set Picture3 bitmap to same as source control.
      Picture3.Picture = Source.Picture
   End If
End Sub
```

Visual Basic Reference

Visual Studio 6.0

DragOver Event

See Also Example Applies To

Occurs when a drag-and-drop operation is in progress. You can use this event to monitor the mouse pointer as it enters, leaves, or rests directly over a valid target. The mouse pointer position determines the target object that receives this event.

Syntax

Private Sub Form_DragOver(*source* **As Control, *x* As Single, *y* As Single, *state* As Integer)**

Private Sub MDIForm_DragOver(*source* **As Control, *x* As Single, *y* As Single, *state* As Integer)**

Private Sub *object*_DragOver([*index* **As Integer,]*source* As Control, *x* As Single, *y* As Single, *state* As Integer)**

The DragOver event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
source	The control being dragged. You can refer to properties and methods in the event procedure with this argument for example, `Source.Visible = False`.
x, *y*	A number that specifies the current horizontal (*x*) and vertical (*y*) position of the mouse pointer within the target form or control. These coordinates are always expressed in terms of the target's coordinate system as set by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties.
state	An integer that corresponds to the transition state of the control being dragged in relation to a target form or control:
	0 = Enter (source control is being dragged within the range of a target).
	1 = Leave (source control is being dragged out of the range of a target).
	2 = Over (source control has moved from one position in the target to another).

Remarks

Use a DragOver event procedure to determine what happens after dragging is initiated and before a control drops onto a target. For example, you can verify a valid target range by highlighting the target (set the **BackColor** or **ForeColor** property from code) or by displaying a special drag pointer (set the **DragIcon** or **MousePointer** property from code).

Use the *state* argument to determine actions at key transition points. For example, you might highlight a possible target when *state* is set to 0 (Enter) and restore the object's previous appearance when *state* is set to 1 (Leave).

When an object receives a DragOver event while *state* is set to 0 (Enter):

- If the source control is dropped on the object, that object receives a DragDrop event.
- If the source control isn't dropped on the object, that object receives another DragOver event when *state* is set to 1 (Leave).

Note Use the **DragMode** property and **Drag** method to specify the way dragging is initiated. For suggested techniques with the *source* argument, see Remarks for the DragDrop event topic.

Visual Basic Reference

DragOver Event Example

This example demonstrates one way to indicate a valid drop target. The pointer changes from the default arrow to a special icon when a **TextBox** control is dragged over a **PictureBox** control. The pointer returns to the default when the source is dragged elsewhere. To try this example, paste the code into the Declarations section of a form that contains a small **TextBox** and a **PictureBox**. Set the **TextBox** control's **DragMode** property to 1, and then press F5 and drag the **TextBox** over the **PictureBox**.

```
Private Sub Picture1_DragOver (Source As Control, X As Single, Y As Single, State As Integer)
   Select Case State
      Case vbEnter
   ' Load icon.
         Source.DragIcon = LoadPicture("ICONS\ARROWS\POINT03.ICO")
      Case vbLeave
         Source.DragIcon = LoadPicture()   ' Unload icon.
   End Select
End Sub

Private Sub Picture1_DragDrop (Source As Control, X As Single, Y As Single)
   Source.DragIcon = LoadPicture()   ' Unload icon.
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

DropDown Event (DateTimePicker Control)

See Also Example Applies To

Occurs when the dropdown calendar is about to drop down.

Syntax

Private Sub *object*_**DropDown**()

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

Visual Basic: Windows Controls

Visual Studio 6.0

DropDown Event (ImageCombo Control)

See Also Example Applies To

Occurs when the list portion of the ImageCombo control is about to drop down.

Syntax

Private Sub *object*_**DropDown( )**

The **DropDown** event syntax has these parts:

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

Remarks

Use a **DropDown** event procedure to make final updates to an **ImageCombo** control list before the user makes a selection. This enables you to add or remove items from the list, change the images displayed for list items, or make other changes to the list, its items or their properties.

Visual Basic: Windows Controls

DropDown Event Example

The following code checks the value of a menu item to see if the status of items should be displayed. If status display is active, the code checks the value of the **Tag** property of each list item and sets the **Image** property accordingly. If status display is inactive, the **Image** for the item is set to an image named "Unchecked."

```
Private Sub ImageCombo1_DropDown()
   If mnuShowStatus.Checked = True Then
      For Each CboItem in ImageCombo1.ComboItems
         Select Case CboItem.Tag
         Case "Locked"
            CboItem.Image = "Padlock"
         Case "Deleted"
            CboItem.Image = "X-mark"
         Case "Checked"
            CboItem.Image = "Checkmark"
         Case Else
            CboItem.Image = "Unchecked"
         End Select
      Next CboItem
   Else
      For Each CboItem in ImageCombo1.ComboItems
         CboItem.Image = "Unchecked"
      Next CboItem
   End If
End Sub
```

In the above code, "Unchecked, "Padlock", "X-mark" and "Checkmark" are key values that indicate particular images in the **ImageList** control associated with the **ImageCombo**.

Visual Basic Reference

Visual Studio 6.0

DropDown Event

See Also Example Applies To

Occurs when the list portion of a **ComboBox** control is about to drop down; this event doesn't occur if a **ComboBox** control's **Style** property is set to 1 (Simple Combo).

Syntax

```
Private Sub object_DropDown([index As Integer])
```

The DropDown event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

Use a DropDown event procedure to make final updates to a **ComboBox** list before the user makes a selection. This enables you to add or remove items from the list using the **AddItem** or **RemoveItem** methods. This flexibility is useful when you want some interplay between controls for example, if what you want to load into a **ComboBox** list depends on what the user selects in an **OptionButton** group.

Visual Basic Reference

DropDown Event Example

This example updates a **ComboBox** control based on the user's selection in an option button group. To try this example, paste the code into the Declarations section of a form that contains a **ComboBox** control and two **OptionButton** controls. Set the **Name** property of both **OptionButton** controls to OptionGroup, and then press F5 and click the **OptionButton** controls. The **ComboBox** control reflects different carriers depending on the **OptionButton** selected.

```
Private Sub Form_Load ()
   Combo1.Text = ""   ' Clear combo box.
End Sub

Private Sub Combo1_DropDown ()
   Combo1.Clear            ' Delete existing items.
   If OptionGroup(0).Value = True Then
      Combo1.AddItem "Gray Goose Express", 0
      Combo1.AddItem "Wild Fargo Carriers", 1
   Else
      Combo1.AddItem "Summit Technologies Overnight"
   End If
End Sub
```