This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

CallbackKeyDown Event

See Also Example Applies To

Occurs when a key is pressed while the cursor is inside a callback field.

Syntax

Private Sub *object*_**CallbackKeyDown([***index* **As Integer],** *KeyCode* **As Integer,** *Shift* **As Integer,** *CallbackField* **As String,** *CallbackDate* **As Date)**

The CallbackKeyDown 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.
KeyCode	A numeric expression that specifies the ASCII key code of the key that was pressed.
Shift	A numeric expression that specifies the state of the SHIFT, CTRL and ALT keys at the time of the event.
CallbackField	A string expression specifying the callback substring.
CallbackDate	A date expression specifying the date value of the control.

Remarks

The CallbackKeyDown event is a variation on the KeyDown event, and can be used to process a callback substring.

See **CustomFormat** Property for more information on callback processing.

The event also occurs when the user presses the up and down arrow keys

Visual Basic Reference

Visual Studio 6.0

Click Event (ActiveX Controls)

See Also Example Applies To

Occurs when the user presses and then releases a mouse button over an object. It can also occur when the value of a control is changed.

Syntax

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

The Click 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

Typically, you attach a Click event procedure to a **CommandButton** control, **Menu** object, or **PictureBox** control to carry out commands and command-like actions. For the other applicable controls, use this event to trigger actions in response to a change in the control.

You can use a control's **Value** property to test the state of the control from code. Clicking a control generates MouseDown and MouseUp events in addition to the Click event. The order in which these three events occur varies from control to control. If the order of events is important in your application, test the control to determine the event order.

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, because the Click event is the first event to trigger between the two. As a result, the mouse click is intercepted by the Click event, so the DblClick event doesn't occur.

Visual Basic: DataCombo/DataList Controls

Visual Studio 6.0

Click Event (DBCombo Control)

See Also Example Applies To

Occurs when the user presses and then releases a mouse button over the **DBCombo** control. This event also occurs by pressing the up or down arrow keys on the keyboard to select an item.

Syntax

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

The Click 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.
Area	An integer expression that specifies where the control was clicked, as described below.

The *Area* parameter can contain the following values:

Constant	Value	Description
dbcAreaButton	0	The user clicked the button on the **DBCombo** control.
dbcAreaEdit	1	The user click in the text box part of the **DBCombo** control.
dbcAreaList	2	The user clicked in the drop-down list part of the **DBCombo** control.

Remarks

The **DBCombo** control does not have a DropDown event like the standard combo box to signal when the user drops down the list portion of the **DBCombo** control. Instead the user can use the *Area* parameter in the Click event to tell the user which area in the **DBCombo** control the user clicked.

Typically, you attach a Click event procedure to a control to carry out commands and command-like actions.

Clicking a control generates MouseDown and MouseUp events in addition to the Click event. When you're attaching event procedures for these related events, be sure that their actions don't conflict. If the order of events is important in your application, test the control to determine the event order.

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

Visual Basic: MSTab Control

Visual Studio 6.0

Click Event (SSTab Control)

See Also Example Applies To

The Click event occurs when the user selects one of the tabs on an **SSTab** control.

Syntax

Private **Sub** *object*_**Click** ([*index* **As Integer**], *previoustab* **As Integer**)

The Click event syntax has these parts:

Part	Description
object	An object expression that evaluates to an **SSTab** control.
Index	An integer that uniquely identifies a control if it is in a control array.
previoustab	A numeric expression that identifies the tab that was previously active.

Remarks

Use the Click event to determine when a user clicks a tab to make it the active tab. When a tab receives a Click event, that tab becomes the active tab and the controls placed on it at design time appear.

With the *previoustab* argument, you can check for changes made when the user clicks another tab.

Use the **Tab** property to determine the current tab.

Visual Basic: MSTab Control

Click Event (SSTab Control) Example

This example saves preferences information from two tabs of an **SSTab** control as soon as the user selects a different tab.

```
Private Sub sstbPrefs_Click(PreviousTab As Integer)
   Dim ThisSetting As String
   Select Case PreviousTab
      Case 0
         If optLoanLen(0) = True Then
            ThisSetting = "Months"
         Else
            ThisSetting = "Years"
         End If
         SaveSetting "LoanSheet", "LoanLength", _
               "Period", ThisSetting
      Case 1
         Dim X As Integer
         For X = 0 To 3
            If optPctsShown(X) = True Then
               SaveSetting "LoanSheet", "InterestRate", _
                "Precision", optPctsShown(X).Tag
               Exit For
            End If
         Next X
   End Select
End Sub
```

Visual Basic Extensibility Reference

Visual Studio 6.0

Click Event (VBA Add-In Object Model)

See Also Example Applies To Specifics

Occurs when the **OnAction** property of a corresponding command bar control is set.

Syntax

Sub *object*_**Click (ByVal** ***ctrl*** **As Object**, **ByRef** ***handled*** **As Boolean**, **ByRef** ***canceldefault*** **As Boolean)**

The Click event syntax has these named arguments:

Part	Description
ctrl	Required; Object. Specifies the object that is the source of the Click event.
handled	Required; Boolean. If **True**, other add-ins should handle the event. If **False**, the action of the command bar item has not been handled.
canceldefault	Required; **Boolean**. If **True**, default behavior is performed unless canceled by a downstream add-in. If **False**, default behavior is not performed unless restored by a downstream add-in.

Remarks

The Click event is specific to the **CommandBarEvents** object. Use a variable declared using the **WithEvents** keyword to receive the Click event for a **CommandBar** control. This variable should be set to the return value of the **CommandBarEvents** property of the **Events** object. The **CommandBarEvents** property takes the **CommandBar** control as an argument. When the **CommandBar** control is clicked (for the variable you declared using the **WithEvents** keyword), the code is executed.

Visual Basic Extensibility Reference

Click Event Example

The following example illustrates how you can set up code for a Click event procedure using **WithEvents** and **Set**. Note that the object reference ce is used in place of the menu name **Tools** in the name of the Click event.

```
Private WithEvents ce As CommandBarEvents

Sub Test()
    Dim c As CommandBarControl
    Set c = Application.VBE.CommandBars("Tools").Controls(1)
    Set ce = Application.VBE.Events.CommandBarEvents(c)
End Sub

Private Sub ce_Click(ByVal CommandBarControl As Object, Handled As Boolean, CancelDefault As Boolean)
    ' Put event-handling code here
End Sub
```

Visual Basic Reference

Visual Studio 6.0

Click Event

See Also Example Applies To

Occurs when the user presses and then releases a mouse button over an object. It can also occur when the value of a control is changed.

For a **Form** object, this event occurs when the user clicks either a blank area or a disabled control. For a control, this event occurs when the user:

- Clicks a control with the left or right mouse button. With a **CheckBox**, **CommandButton**, **Listbox**, or **OptionButton** control, the Click event occurs only when the user clicks the left mouse button.
- Selects an item in a **ComboBox** or **ListBox** control, either by pressing the arrow keys or by clicking the mouse button.
- Presses the SPACEBAR when a **CommandButton**, **OptionButton**, or **CheckBox** control has the focus.
- Presses ENTER when a form has a **CommandButton** control with its **Default** property set to **True**.
- Presses ESC when a form has a Cancel button a **CommandButton** control with its **Cancel** property set to **True**.
- Presses an access key for a control. For example, if the caption of a **CommandButton** control is "&Go", pressing ALT+G triggers the event.

You can also trigger the Click event in code by:

- Setting a **CommandButton** control's **Value** property to **True**.
- Setting an **OptionButton** control's **Value** property to **True**.
- Changing a **CheckBox** control's **Value** property setting.

Syntax

```
Private Sub Form_Click( )

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

The Click 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

Typically, you attach a Click event procedure to a **CommandButton** control, **Menu** object, or **PictureBox** control to carry out commands and command-like actions. For the other applicable controls, use this event to trigger actions in response to a change in the control.

You can use a control's **Value** property to test the state of the control from code. Clicking a control generates MouseDown and MouseUp events in addition to the Click event. The order in which these three events occur varies from control to control. For example, for **ListBox** and **CommandButton** controls, the events occur in this order: MouseDown, Click, MouseUp. But for **FileListBox**, **Label**, or **PictureBox** controls, the events occur in this order: MouseDown, MouseUp, and Click. When you're attaching event procedures for these related events, be sure that their actions don't conflict. If the order of events is important in your application, test the control to determine the event order.

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 DblClick event will never trigger, because the Click event is the first event to trigger between the two. As a result, the mouse click is intercepted by the Click event, so the DblClick event doesn't occur.

Visual Basic Reference

Click Event Example

In this example, each time a **PictureBox** control is clicked it moves diagonally across a form. To try this example, paste the code into the Declarations section of a form that contains a **PictureBox** control positioned at the lower-left corner of the form, and then press F5 and click the **PictureBox**.

```
Private Sub Picture1_Click ()
   Picture1.Move Picture1.Left + 750, Picture1.Top - 550
End Sub
```

Visual Basic: Winsock Control

Visual Studio 6.0

Close Event

See Also Example Applies To

Occurs when the remote computer closes the connection. Applications should use the **Close** method to correctly close a TCP connection.

Syntax

*object*_**Close( )**

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

Arguments

None

Visual Basic Reference

Visual Studio 6.0

CloseQuery Event

See Also Example Applies To

Occurs after you close a query window.

Syntax

object.**CloseQuery(***value***)**

Parameters

The **CloseQuery** 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 most recent DECommand object that was edited in the query window.

Visual Basic: Windows Controls

Visual Studio 6.0

CloseUp Event

See Also Example Applies To

Occurs when the drop-down calendar is closed.

Syntax

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

The CloseUp 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 CloseUp event can be used to respond to the user closing the drop-down calendar.

Visual Basic: DataGrid Control

Visual Studio 6.0

ColEdit Event

See Also Example Applies To

Occurs when a cell first enters edit mode by typing a character.

Syntax

Private Sub *object*_**ColEdit(**[*index* **As Integer**,] **ByVal** *colindex* **As Integer)**

The ColEdit 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 identifies a control if it is in a control array.
colindex	An integer that identifies the column being edited.

Remarks

If a floating editor marquee is not in use, this event also occurs when the user clicks the current cell or double clicks another cell.

The ColEdit event immediately follows the BeforeColEdit event only when the latter is not canceled.

When the user completes editing within a grid cell, as when tabbing to another column in the same row, pressing the ENTER key, or clicking on another cell, the BeforeColUpdate and AfterColUpdate events are executed if the data has been changed. The AfterColEdit event is then fired to indicate that editing is completed.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

Collapse Event (MSHFlexGrid)

SeeAlso Example Applies To

Occurs when the user collapses a row within the grid. The **Col** and **Row** properties of the **MSHFlexGrid** contain the cell used to collapse the band.

Syntax

Private Sub *object_* **Collapse(***Boolean***)**

The Collapse event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
Boolean	A Boolean expression. If you set **Cancel** to **True**, the collapse is cancelled.

Visual Basic: Windows Controls

Visual Studio 6.0

Collapse Event (TreeView Control)

See Also Example Applies To

Generated when any **Node** object in a **TreeView** control is collapsed.

Syntax

```
Private Sub object_Collapse(ByVal node As Node)
```

The Collapse event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
node	A reference to the clicked **Node** object.

Remarks

The Collapse event occurs before the standard Click event.

There are three methods of collapsing a **Node**: by setting the **Node** object's **Expanded** property to **False**, by double-clicking a **Node** object, and by clicking a plus/minus image when the **TreeView** control's **Style** property is set to a style that includes plus/minus images. All of these methods generate the Collapse event.

The event passes a reference to the collapsed **Node** object. The reference can validate an action, as in the following example:

```
Private Sub TreeView1_Collapse(ByVal Node As Node)
   If Node.Index = 1 Then
      Node.Expanded = True    ' Expand the node again.
   End If
End Sub
```

Visual Basic: Windows Controls

Collapse Event (TreeView Control) Example

This example adds one **Node** object, with several child nodes, to a **TreeView** control. When the user collapses a **Node,** the code checks to see how many children the **Node** has. If it has more than one child, the **Node** is re-expanded. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and double-click a **Node** to collapse it and generate the event.

```
Private Sub Form_Load()
   TreeView1.Style = tvwTreelinesPlusMinusText ' Style 6.
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,"DV","Da Vinci")
   Set nodX = TreeView1.Nodes.Add("DV",tvwChild,"T","Titian")
   Set nodX = TreeView1.Nodes.Add("T",tvwChild,"R","Rembrandt")
   Set nodX = TreeView1.Nodes.Add("R",tvwChild,,"Goya")
   Set nodX = TreeView1.Nodes.Add("R",tvwChild,,"David")
   nodX.EnsureVisible   ' Show all nodes.
End Sub

Private Sub TreeView1_Collapse(ByVal Node As Node)
   ' If the Node has more than one child node,
   ' keep the node expanded.
   Select Case Node.Children
      Case Is > 1
         Node.Expanded = True
   End Select
End Sub
```

Visual Basic: DataGrid Control

Visual Studio 6.0

ColResize Event

See Also Example Applies To

Occurs when a user resizes a column of a **DataGrid** control.

Syntax

Private Sub *object*_**ColResize (**[*index* **As Integer**,] *colindex* **As Integer**, *cancel* **As Integer)**

The ColResize 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 identifies a control if it is in a control array.
colindex	An integer that identifies the column.
cancel	A Boolean expression that determines whether a column is resized, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	Cancels the change, restores column to its original width.
False	(Default) Continues with width change.

Remarks

When the user resizes a column, the ColResize event is triggered. Your event procedure can accept the change, alter the degree of change, or cancel the change completely.

If you set the *cancel* argument to **True**, the column width is restored. To alter the degree of change, set the **Width** property of the **Column** object to the desired value.

Executing the **Refresh** method within the procedure causes the control to be repainted even if the *cancel* argument is **True**.

Visual Basic: DataGrid Control

ColResize Event Example

This example resizes all the columns to the size of the first column if the user sizes the first column.

```
Private Sub DataGrid1_ColResize (ColIndex As Integer, Cancel As Integer)
   Dim nCol As Column
   If ColIndex = 1 Then
      For Each nCol In DataGrid1.Columns
         nCol.Width = DataGrid.Columns(1).Width
      Next
   End If
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

ColumnClick Event

See Also Example Applies To

Occurs when a **ColumnHeader** object in a **ListView** control is clicked. Only available in Report view.

Syntax

```
Private Sub object_ColumnClick(ByVal columnheader As ColumnHeader)
```

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

Part	Description
object	An object expression that evaluates to a **ListView** control.
columnheader	A reference to the **ColumnHeader** object that was clicked.

Remarks

The **Sorted**, **SortKey**, and **SortOrder** properties are commonly used in code to sort the **ListItem** objects in the clicked column.

Visual Basic: Windows Controls

SortKey, SortOrder, Sorted Properties, ColumnClick Event Example

This example adds three **ColumnHeader** objects to a **ListView** control and populates the control with the Publishers records of the Biblio.mdb database. An array of two **OptionButton** controls offers the two choices for sorting records. When you click on a **ColumnHeader**, the **ListView** control is sorted according to the **SortOrder** property, as determined by the **OptionButtons**. To try the example, place a **ListView** and a control array of two **OptionButton** controls on a form and paste the code into the form's Declarations section. Run the example and click on the **ColumnHeaders** to sort, and click on the **OptionButton** to switch the **SortOrder** property.

Note The example will not run unless you add a reference to the Microsoft DAO 3.51 Object Library by using the References command on the Tools menu.

```
Private Sub Option1_Click(Index as Integer)
   ' These OptionButtons offer two choices: Ascending (Index 0),
   ' and Descending (Index 1). Clicking on one of these
   ' sets the SortOrder for the ListView control.
   ListView1.SortOrder = Index
   ListView1.Sorted = True ' Sort the List.
End Sub

Private Sub Form_Load()
   ' Create an object variable for the ColumnHeader object.
   Dim clmX As ColumnHeader
   ' Add ColumnHeaders. The width of the columns is the width
   ' of the control divided by the number of ColumnHeader objects.
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Company", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Address", ListView1.Width / 3)
   Set clmX = ListView1.ColumnHeaders. _
   Add(, , "Phone", ListView1.Width / 3)

   ListView1.BorderStyle = ccFixedSingle ' Set BorderStyle property.
   ListView1.View = lvwReport ' Set View property to Report.

   ' Label OptionButton controls with SortOrder options.
      Option1(0).Caption = "Ascending (A-Z)"
      Option1(1).Caption = "Descending (Z-A)"
      ListView1.SortOrder = lvwAscending ' Sort ascending.

   ' Create object variables for the Data Access objects.
   Dim myDb As Database, myRs As Recordset
   ' Set the Database to the BIBLIO.MDB database.
   Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")
   ' Set the recordset to the Publishers table.
   Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

   ' Create a variable to add ListItem objects.
   Dim itmX As ListItem
```

```
    ' While the record is not the last record, add a ListItem object.
    ' Use the Name field for the ListItem object's text.
    ' Use the Address field for the ListItem object's subitem(1).
    ' Use the Phone field for the ListItem object's subitem(2).

    While Not myRs.EOF
       Set itmX = ListView1.ListItems.Add(, , CStr(myRs!Name))

       ' If the Address field is not Null, set subitem 1 to the field.
       If Not IsNull(myRs!Address) Then
          itmX.SubItems(1) = CStr(myRs!Address)  ' Address field.
       End If

       ' If the Phone field is not Null, set subitem 2 to the field.
       If Not IsNull(myRs!Telephone) Then
          itmX.SubItems(2) = myRs!Telephone  ' Phone field.
       End If

       myRs.MoveNext   ' Move to next record.
    Wend
End Sub

Private Sub ListView1_ColumnClick(ByVal ColumnHeader As ColumnHeader)
    ' When a ColumnHeader object is clicked, the ListView control is
    ' sorted by the subitems of that column.
    ' Set the SortKey to the Index of the ColumnHeader - 1
    ListView1.SortKey = ColumnHeader.Index - 1
    ' Set Sorted to True to sort the list.
    ListView1.Sorted = True
End Sub
```

Visual Basic: RDO Data Control

Visual Studio 6.0

CommitTrans Event

See Also Example Applies To

Occurs after the **CommitTrans** method has completed.

Syntax

Private Sub *object*.**CommitTrans**( )

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

Remarks

This event is raised after a **CommitTrans** method has been executed. The developer can respond to this event to synchronize some other process with the transaction.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

Compare Event

See Also Example Applies To

Occurs when the **Sort** property for the **MSHFlexGrid** is set to Custom Sort (9), so the user can customize the sort process.

Syntax

Private Sub *object*_**Compare(***row1*, *row2*, *cmp***)**

The Compare event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
row1	A Long integer that specifies the first row in a pair of rows being compared.
row2	A Long integer that specifies the second row in a pair of rows being compared.
cmp	An integer that represents the sort order of each pair, as described in Settings.

Settings

The event handler must compare *row1* and *row2* and set *cmp* to:

Setting	Description
1	If *row1* should appear *before row2*.
0	If both rows are equal or either row can appear before the other.
1	If *row1* should appear *after row2*.

Remarks

When the **Sort** property is set to 9 (Custom Sort), the Compare event occurs once for each pair of rows in the **MSHFlexGrid**. As the Compare event uses row numbers instead of text values, you can compare any property value for that row, including **RowData**.

Note While custom sorts are slower than built-in sorts, they provide the flexibility to sort a row by any column or using any cell property.

Visual Basic: SysInfo Control

Visual Studio 6.0

ConfigChangeCancelled Event

See Also Example Applies To

Occurs when the operating system sends a message to all applications that a change to the hardware profile was cancelled.

Syntax

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

The ConfigChangeCancelled 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

You should use this event to clean up any processing begun in the QueryChangeConfig event procedure.

Visual Basic: SysInfo Control

Visual Studio 6.0

ConfigChanged Event

See Also Example Applies To

Occurs when the hardware profile on the system has changed.

Syntax

Private Sub *object*_**ConfigChanged(**[*index* **As Integer,**] **ByVal** *oldconfignum* **As Long**, **ByVal** *newconfignum* **As Long)**

The ConfigChanged 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.
oldconfignum	The key name in the Windows registry for the old system configuration.
newconfignum	The key name in the Windows registry for the new system configuration.

Remarks

For docking and undocking a portable computer with a docking station, this event occurs after the system resumes operation.

Visual Basic: RDO Data Control

Visual Studio 6.0

Connect Event

See Also Example Applies To

Occurs after a connection is established to the server.

Private Sub *object*.**Connect**(*ErrorOccurred* **As Boolean**)

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
ErrorOccurred	A **Boolean** expression that determines whether the connection was successful, as described in Settings.

Settings

The *ErrorOccurred* argument will be set to one of the following values:

Value	Description
True	The connection failed.
False	The connection succeeded.

Remarks

You can catch the Connect event and do any kind of initial queries required on a new connection, such as verifying the version of the database against the version of the client or setting a default database not established in the connect string. You can also check for errors or messages returned during the process of opening the connection or perhaps simply clear the **rdoErrors** collection of informational messages.

Visual Basic: Winsock Control

Visual Studio 6.0

Connect Event (Winsock Control)

See Also Example Applies To

Occurs when a Connect operation is completed.

Syntax

object.Connect()

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

Remarks

Use the Connect event to confirm when a connection has been made successfully.

Visual Basic: Winsock Control

Visual Studio 6.0

ConnectionRequest Event

See Also Example Applies To

Occurs when a remote machine requests a connection.

- For TCP server applications only. The event is activated when there is an incoming connection request. **RemoteHostIP** and **RemotePort** properties store the information about the client after the event is activated.

Syntax

*object*_**ConnectionRequest** (*requestID* **As Long**)

The ConnectionRequest event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
requestID	The incoming connection request identifier. This argument should be passed to the **Accept** method on the second control instance.

Remarks

The server can decide whether or not to accept the connection. If the incoming connection is not accepted, the peer (client) will get the Close event. Use the **Accept** method (on a new control instance) to accept an incoming connection.

Visual Basic: Winsock Control

Accept Method, ConnectionRequest Event Example

The example shows the code necessary to connect a **Winsock** control using the TCP protocol. The code runs on the machine that is accepting the connection request. The **RequestID** parameter identifies the request. This is passed to the **Accept** method which accepts the particular request.

```
Private Sub WinsockTCP_ConnectionRequest _
(requestID As Long)
   If Winsock1.State <> sckClosed Then Winsock1.Close
   Winsock.Accept requestID
End Sub
```

Visual Basic: DataRepeater Control

Visual Studio 6.0

CurrentRecordChanged Event

See Also Example Applies To

Occurs when the current record changes to a different record.

Syntax

Private Sub *object*_**CurrentRecordChanged()**

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

Remarks

The user can change the current record by clicking any record on the control, or by using the forward, backward, move first, and move last buttons on a data control.

Visual Basic: DataRepeater Control CurrentRecordChanged Event Example

The example stores the bookmark of the current record using the **CurrentRecord** property.

```
Option Explicit
Option Base 1
Private mRecords(10) As Variant ' Array for bookmarks.
                                ' We only store last 10.

Private Sub DataRepeater1_CurrentRecordChanged()
    If intC = 10 Then intC = 1
    mRecords(intC) = DataRepeater1.CurrentRecord
    intC = intC + 1
End Sub
```

Visual Basic Reference

Visual Studio 6.0

Change Event (ActiveX Controls)

See Also Example Applies To

Indicates the contents of a control or a property value have changed. How and when this event occurs varies with the control.

Syntax

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

The Change 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 Change event procedure can synchronize or coordinate data display among controls. For example, you can use a scroll bar's Change event procedure to update the scroll bar's **Value** property setting in a **TextBox** control. Or you can use a Change event procedure to display data and formulas in a work area and results in another area.

Note A Change event procedure can sometimes cause a cascading event. This occurs when the control's Change event alters the control's contents, for example, by setting a property in code that determines the control's value, such as the **Text** property setting for a **TextBox** control. To prevent a cascading event:

- If possible, avoid writing a Change event procedure for a control that alters that control's contents. If you do write such a procedure, be sure to set a flag that prevents further changes while the current change is in progress.
- Avoid creating two or more controls whose Change event procedures affect each other, for example, two **TextBox** controls that update each other during their Change events.
- Avoid using a **MsgBox** function or statement in this event for **HScrollBar** and **VScrollBar** controls.

Visual Basic: Windows Controls

Visual Studio 6.0

Change Event (ToolBar, Slider Controls)

See Also Example Applies To

Indicates that the contents of a control have changed. How and when this event occurs varies with the control.

Syntax

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

The Change event syntax has these parts:

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

Remarks

- Slider generated when the **Value** property changes, either through code, or when the user moves the control's slider.
- Toolbar generated after the end user customizes a Toolbar control's toolbar using the Customize Toolbar dialog box.

The Change event procedure can synchronize or coordinate data display among controls. For example, you can use a Slider control's Change event procedure to update the control's **Value** property setting in a TextBox control. Or you could use a Change event procedure to display data and formulas in a work area and results in another area.

Note A Change event procedure can sometimes cause a cascading event. This occurs when the control's Change event alters the control's contents by setting a property in code that determines the control's value, such as the **Text** property setting for a TextBox control. To prevent a cascading event:

- If possible, avoid writing a Change event procedure for a control that alters that control's contents. If you do write such a procedure, be sure to set a flag that prevents further changes while the current change is in progress.
- Avoid creating two or more controls whose Change event procedures affect each other, for example, two TextBox controls that update each other during their Change events.

Visual Basic Reference

Change Event Example

This example displays the numeric setting of a horizontal scroll bar's **Value** property in a **TextBox** control. To try this example, create a form with a **TextBox** control and an **HScrollBar** control and then paste the code into the Declarations section of a form that contains a horizontal scroll bar (**HScrollBar** control) and a **TextBox** control. Press F5 and click the horizontal scroll bar.

```
Private Sub Form_Load ()
   HScroll1.Min = 0               ' Set Minimum.
   HScroll1.Max = 1000            ' Set Maximum.
   HScroll1.LargeChange = 100   ' Set LargeChange.
   HScroll1.SmallChange = 1      ' Set SmallChange.
End Sub

Private Sub HScroll1_Change ()
   Text1.Text = HScroll1.Value
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

Change Event (UpDown Control)

See Also Example Applies To

This event occurs when the **Value** property is changed.

Syntax

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

The Change 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 Change event occurs whenever the **Value** property changes. The **Value** property can change through code, by clicking the arrow buttons, or by changing the value in a buddy control when the **SyncBuddy** property is **True**.

Visual Basic Reference

Visual Studio 6.0

Change Event

See Also Example Applies To

Indicates the contents of a control have changed. How and when this event occurs varies with the control:

- **ComboBox** changes the text in the text box portion of the control. Occurs only if the **Style** property is set to 0 (Dropdown Combo) or 1 (Simple Combo) and the user changes the text or you change the **Text** property setting through code.
- **DirListBox** changes the selected directory. Occurs when the user double-clicks a new directory or when you change the **Path** property setting through code.
- **DriveListBox** changes the selected drive. Occurs when the user selects a new drive or when you change the **Drive** property setting through code.
- **HScrollBar** and **VScrollBar** (horizontal and vertical scroll bars) move the scroll box portion of the scroll bar. Occurs when the user scrolls or when you change the **Value** property setting through code.
- **Label** changes the contents of the **Label**. Occurs when a DDE link updates data or when you change the **Caption** property setting through code.
- **PictureBox** changes the contents of the **PictureBox**. Occurs when a DDE link updates data or when you change the **Picture** property setting through code.
- **TextBox** changes the contents of the text box. Occurs when a DDE link updates data, when a user changes the text, or when you change the **Text** property setting through code.

Syntax

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

The Change 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 Change event procedure can synchronize or coordinate data display among controls. For example, you can use a scroll bar's Change event procedure to update the scroll bar's **Value** property setting in a **TextBox** control. Or you can use a

Change event procedure to display data and formulas in a work area and results in another area.

Change event procedures are also useful for updating properties in file-system controls (**DirListBox**, **DriveListBox**, and **FileListBox**). For example, you can update the **Path** property setting for a **DirListBox** control to reflect a change in a **DriveListBox** control's **Drive** property setting.

Note A Change event procedure can sometimes cause a cascading event. This occurs when the control's Change event alters the control's contents, for example, by setting a property in code that determines the control's value, such as the **Text** property setting for a **TextBox** control. To prevent a cascading event:

- If possible, avoid writing a Change event procedure for a control that alters that control's contents. If you do write such a procedure, be sure to set a flag that prevents further changes while the current change is in progress.
- Avoid creating two or more controls whose Change event procedures affect each other, for example, two **TextBox** controls that update each other during their Change events.
- Avoid using a **MsgBox** function or statement in this event for **HScrollBar** and **VScrollBar** controls.

Visual Basic Reference

Change Event Example

This example displays the numeric setting of a horizontal scroll bar's **Value** property in a **TextBox** control. To try this example, create a form with a **TextBox** control and an **HScrollBar** control and then paste the code into the Declarations section of a form that contains a horizontal scroll bar (**HScrollBar** control) and a **TextBox** control. Press F5 and click the horizontal scroll bar.

```
Private Sub Form_Load ()
   HScroll1.Min = 0               ' Set Minimum.
   HScroll1.Max = 1000            ' Set Maximum.
   HScroll1.LargeChange = 100   ' Set LargeChange.
   HScroll1.SmallChange = 1      ' Set SmallChange.
End Sub

Private Sub HScroll1_Change ()
   Text1.Text = HScroll1.Value
End Sub
```

Visual Basic Reference

Visual Studio 6.0

Changed Event

See Also Example Applies To

Occurs when a property of the **StdDataFormat** object changes.

Syntax

*object*_**Changed**()

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

Remarks

To use the event, declare an object variable in the Declarations section, as type **StdDataFormat**, using the WithEvents keyword as shown below:

```
Option Explicit
Private WithEvents fmtMyFormat As StdDataFormat

Private Sub fmtMyFormat_Changed()
   ' Handle change event here.
End Sub
```

Visual Basic Reference

Changed Event Example

The example prints out the number of times the Changed event has occurred. The example assumes that a **TextBox** control has been bound to a data source, such as the **ADO Data Control**, at design time.

```
Option Explicit
Private WithEvents fmtMyFormat As StdDataFormat

Private Sub Form_Load()
   Set fmtMyFormat = New StdDataFormat
   fmtMyFormat.Format = "Long Date"
   Set Text1.DataFormat = fmtMyFormat
End Sub

Private Sub Command1_Click()
   fmtMyFormat.FirstDayOfWeek = fmtMonday
End Sub

Private Sub fmtMyFormat_Changed()
    Static fmtMyFormatChange As Integer
    fmtMyFormatChange = fmtMyFormatChange + 1
    Debug.Print "CHANGED", fmtMyFormatChange
End Sub
```

Visual Studio 6.0

Visual Basic: MSChart Control

ChartActivated Event

See Also Example Applies To

Occurs when the user double clicks the Microsoft Chart control, but not on a specific element in the chart.

Syntax

Private Sub *object*_**ChartActivated** (*mouseFlags* **As Integer**, *cancel* **As Integer**)

The ChartActivated event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
mouseFlags	Integer. Indicates whether a key is held down when the mouse button is clicked, as described in Settings.
cancel	Integer. This argument is not used at this time.

Settings

The event handler determines if a key is held down when the mouse button is clicked and sets *mouseFlags* to:

Constants	Description
VtChMouseFlagShiftKeyDown	If the SHIFT key is held down.
VtChMouseFlagControlKeyDown	If the CONTROL key is held down.

Visual Studio 6.0

Visual Basic: MSChart Control

ChartSelected Event

See Also Example Applies To

Occurs when the user clicks the Microsoft Chart control, but not on a specific element in the chart.

Syntax

Private Sub *object*_**ChartSelected** (*mouseFlags* **As Integer**, *cancel* **As Integer**)

The ChartSelected event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
mouseFlags	Integer. Indicates whether a key is held down when the mouse button is clicked, as described in Settings.
cancel	Integer. This argument is not used at this time.

Settings

The event handler determines if a key is held down when the mouse button is clicked and sets *mouseFlags* to:

Constants	Description
VtChMouseFlagShiftKeyDown	If the SHIFT key is held down.
VtChMouseFlagControlKeyDown	If the CONTROL key is held down.

Visual Studio 6.0

Visual Basic: MSChart Control

ChartUpdated Event

See Also Example Applies To

Occurs when the chart has changed.

Syntax

Private Sub *object*_**ChartUpdated** (*updateFlags* **As Integer**)

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
updateFlags	Integer. Provides information about the update of the chart, 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.