This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

BeforeClick Event

See Also Example Applies To

Generated when a **Tab** object in a **TabStrip** control is clicked, or a **Tab** object's **Selected** setting has changed.

Syntax

Private Sub *object*_**BeforeClick(***cancel* **As Integer)**

The BeforeClick event syntax has these parts:

Part	Description
object	An object expression that evaluates to a **TabStrip** control.
cancel	Evaluates to an integer with values of 0 (False) and -1 (True). The initial value is 0.

Remarks

Use the BeforeClick event to validate the information on the old **Tab** object before actually generating a Click event that selects the new **Tab** object. Setting the *cancel* argument to **True** allows you to stop a change to the new selection.

Note Setting the *cancel* argument to **True** prevents the focus from switching to another tab but doesn't stop the Click event from occurring.

Note If you use the **MsgBox** or **InputBox** functions during the BeforeClick event procedure, the **TabStrip** control will not receive a Click event, regardless of the setting of the *cancel* argument.

© 2018 Microsoft

Visual Basic: Windows Controls

BeforeClick Event Example

This example uses the BeforeClick event to demonstrate how to prevent a user from switching to another tab. This is useful when you want to verify information on the current tab before displaying the newly selected tab.

To try this example, place a **TabStrip** control and a two-element **Frame** control array on the form (set the BorderStyle properties to None). In the first **Frame** control, add a **CheckBox** control and in the second, add a **TextBox**. Paste the following code into the Load event of the Form object, and run the program. Click the tab labeled Text after you select/deselect the CheckBox on the tab labeled Check.

```
Private Sub Form_Load()
Dim i As Integer
Dim Tabx As Object
' Sets the caption of the first tab to "Check."
TabStrip1.Tabs(1).Caption = "Check"
' Adds a second tab with "Text" as its caption.
Set Tabx = TabStrip1.Tabs.Add(2, , "Text")
' Labels the checkbox.
Check1.Caption = "Cancel tab switch"
   ' Aligns the Frames with the internal area
   ' of the Tabstrip Control.
   For i = 0 To 1
      Frame1(i).Left = TabStrip1.ClientLeft
      Frame1(i).Top = TabStrip1.ClientTop
      Frame1(i).Height = TabStrip1.ClientHeight
      Frame1(i).Width = TabStrip1.ClientWidth
   Next
   ' Puts the first tab's Frame container on top.
   Frame1(0).ZOrder 0
End Sub

' The BeforeClick event verifies the check box value
' to determine whether to proceed with the Click event.
Private Sub TabStrip1_BeforeClick(Cancel As Integer)
   If TabStrip1.Tabs(1).Selected Then
      If Check1.Value = 1 Then Cancel = True
   End If
End Sub

Private Sub TabStrip1_Click()
   Frame1(TabStrip1.SelectedItem.Index-1).ZOrder 0
End Sub
```

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

BeforeColEdit Event

See Also Example Applies To

Occurs just before the user enters edit mode by typing a character.

Syntax

Private Sub *object*_**BeforeColEdit(**[*index* **As Integer**,] **ByVal** *colindex* **As Integer**, **ByVal** *keyascii* **As Integer**, *cancel* **As Integer)**

The BeforeColEdit 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 to be edited.
keyascii	An integer representing the ANSI key code of the character typed by the user to initiate editing, or 0 if the user initiated editing by clicking the mouse. KeyAscii is passed by value, not by reference; you cannot change its value to initiate editing with a different character.
cancel	An integer that may be set to **True** to prevent the user from editing the cell, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	The cell will not enter edit mode
False	(Default) The ColEdit event is fired immediately, followed by the Change and KeyUp events for the KeyAscii argument, if non-zero.

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.

Use this event to control the editability of cells on a per-cell basis, or to translate the initial keystroke into a default value.

Note The *keyascii* argument can only be 0 if a floating editor marquee is not in use.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

BeforeColUpdate Event

See Also Example Applies To

Occurs after editing is completed in a cell, but before data is moved from the cell to the **DataGrid** control's copy buffer.

Syntax

Private Sub *object*_**BeforeColUpdate (**[*index* **As Integer**,] *colindex* **As Integer**, *oldvalue* **As Variant**, *cancel* **As Integer)**

The BeforeColUpdate 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.
oldvalue	A value that contains the value contained in the cell prior to the change.
cancel	A Boolean expression expression that specifies whether the change occurs, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	Cancels the change, restores cell to *oldvalue*, and restores focus to the control.
False	(Default) Continues with change and permits change of focus.

Remarks

The data specified by the *oldvalue* argument moves from the cell to the control's copy buffer when a user completes editing within a cell, as when tabbing to another column in the same row, pressing ENTER, or changing focus away from the cell. Before the data has been moved from the cell into the control's copy buffer, the BeforeColUpdate event is triggered. This

event gives your application an opportunity to check the individual grid cells before they are committed to the control's copy buffer.

If your event procedures set the *cancel* argument to **True**, the previous value is restored in the cell and focus remains on the control and the AfterColUpdate event is not triggered.

To restore *oldvalue* in the cell and permit the user to move focus off of the cell, set *cancel* to **False** and set the cell to *oldvalue* as follows:

```
Cancel = False
DataGrid1.Columns(ColIndex).Value = OldValue
```

The AfterColUpdate event occurs after the BeforeColUpdate event.

By setting the *cancel* argument to **True**, the user can not move the focus from the control until the application determines that the data can be safely moved back to the control's copy buffer.

© 2018 Microsoft

Visual Basic: DataGrid Control

BeforeColUpdate Event Example

This example checks to make sure that the value the user has typed in is within a certain range; otherwise it disables the update.

```
Private Sub DataGrid1.BeforeColUpdate (ColIndex As Long, OldValue As Variant, Cancel As Integer)
   If ColIndex = 1 Then
      If DataGrid1.Columns(1).Value < Now Then
         Cancel = True
         MsgBox "You must enter a date that is later than today."
      End If
   End If
End Sub
```

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: RDO Data Control

Visual Studio 6.0

BeforeConnect Event

See Also Example Applies To

Occurs just before RDO calls the ODBC API **SQLDriverConnect** function to establish a connection to the server.

Syntax

Private Sub *object*.**BeforeConnect**(*ConnectString* as **String**, *Prompt* as **Variant**)

The BeforeConnect event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
ConnectString	A **Variant** expression that evaluates to a connect string used to provide connect parameters for the ODBC **SQLDriverConnect** function.
Prompt	Determines how the user should be prompted.

Remarks

The BeforeConnect event is fired just before RDO calls the ODBC API **SQLDriverConnect** function to establish a connection to the server. This event gives your code an opportunity to provide custom prompting, or just provide or capture connection information.

The ***ConnectString*** parameter is the ODBC connect string RDO will pass to the ODBC API **SQLDriverConnect** function. This string can be changed during this event, and RDO will use the changed value. For example, your code can provide additional parameters, or change existing parameters of the connect string.

The ***Prompt*** parameter is the ODBC prompting constant (see the **Prompt** property). This parameter will default to the value of the ***Prompt*** parameter passed in the **OpenConnection** or **EstablishConnection** methods. The developer may change this value, and RDO will use the new value when calling **SQLDriverConnect**.

© 2018 Microsoft

Visual Basic: RDO Data Control

RDO Events Example

This example illustrates several of the Remote Data Object (RDO) event handlers. The code establishes event variables and handlers to trap connection and query events. To help illustrate use of the BeforeConnect event, the code concatenates a workstation ID value and the current time to the end of the connect string. This permits identification of the specific connection at the server. After establishing the connection, the code executes a query that takes an fairly long time to execute the query is designed to run for about a minute. Because a 5 second QueryTimeout value is set, the QueryTimeout event should fire unless the query returns before 5 seconds has elapsed. Notice that the query itself is run asynchronously and the code does not poll for completion of the query. In this case the code simply waits for the QueryComplete or QueryTimeout events to fire indicating that the query is finished. The code also permits you to request another 5 seconds of waiting time.

Note that to make this example work correctly, you must do a number of things first, including setting references to the Remote Data Objects and Common Dialog libraries, adding a **CommandButton** and a **Timer** control to a form, plus you must change the ODBC connect string in the Form_Load() event to point to a valid server.

```
Option Explicit
Private WithEvents cn As rdoConnection
Private WithEvents EngEv As rdoEngine
Dim er As rdoError
Dim strConnect As String
Dim rs As rdoResultset
Dim TimeStart As Single
Dim clock As Integer
Dim ShowClock As Integer
Dim QueryComplete As Integer
Dim InfoMsg As String
Dim Connected As Boolean
Dim ans As Integer

Private Sub EngEv_InfoMessage()
    InfoMsg = "For your information..." _
    & " the following message" _
        & " was returned by the server." & vbCrLf
    For Each er In rdoErrors
        InfoMsg = InfoMsg & er.Number _
    & " - " & er.Description & vbCrLf
    Next

End Sub
Private Sub cn_BeforeConnect( _
    ConnectString As String, Prompt As Variant)
  InfoMsg = "About to connect to:" & ConnectString _
    & " - " & Prompt
  ConnectString = ConnectString & ";WSID=" _
    & "EventTest" & Time$ & ";"
End Sub

Private Sub cn_Connect(ByVal ErrorOccurred As Boolean)
  'Fires once connected.
  Connected = True
End Sub
```

```
Private Sub cn_Disconnect() 'Fires when disconnected
  Connected = False
End Sub

Private Sub cn_QueryComplete(ByVal Query As _
  RDO.rdoQuery, ByVal ErrorOccurred As Boolean)
  Timer1.Enabled = False
  QueryComplete = vbChecked
  RunButton.Enabled = True
  Beep

  MsgBox "Query Done"
End Sub

Private Sub cn_QueryTimeout(ByVal Query As _
  RDO.rdoQuery, Cancel As Boolean)
    ans = MsgBox("The query did not complete " _
    & "in the time allocated. " _
    & "Press Cancel to abandon the query " _
    & "or Retry to keep working.", _
        vbRetryCancel + vbQuestion, "Query Timed Out")
    If ans = vbRetry Then
        Cancel = False
        QueryComplete = vbGrayed
    Else
        Timer1.Enabled = False
        QueryComplete = vbChecked
    End If
End Sub

Private Sub MenufileExit_Click()
    cn.Close
    Unload Form1
End Sub

Private Sub RunButton_Click()
    RunButton.Enabled = False
    On Error GoTo C1EH
    QueryComplete = vbGrayed
    Timer1.Enabled = True
    Set rs = cn.OpenResultset( _
        "execute VeryLongProcedure", _
        rdOpenKeyset, rdConcurValues, rdAsyncEnable)
    TimeStart = Timer
QuitRun:
Exit Sub
C1EH:
    Debug.Print Err, Error
        InfoMsg = "Error:.. the following error" _
        & " was returned by the server." & vbCrLf
    For Each er In rdoErrors
        InfoMsg = InfoMsg & er.Number _
        & " - " & er.Description & vbCrLf
    Next
    MsgBox "Query Failed to run"
    Timer1.Enabled = False
    Resume QuitRun

End Sub
```

```
Private Sub Form_Load()
On Error GoTo FLeh
Set EngEv = rdoEngine
Set cn = New rdoConnection
Show
    With cn
        .Connect = "UID=;PWD=;database=Workdb;" _
            & "Server=SEQUEL;" _
                & "driver={SQL Server};DSN='';"
        .QueryTimeout = 5
        .CursorDriver = rdUseClientBatch
        .EstablishConnection rdDriverNoPrompt
    End With
Exit Sub

FLeh:
    Debug.Print Err, Error
    For Each er In rdoErrors
        Debug.Print er.Description
    Next
    Stop
    Resume
End Sub

Private Sub Timer1_Timer()
    Static ot As Integer
    ' Display number of seconds
    ShowClock = Int(Timer - TimeStart)
    If ShowClock = ot Then Form1.Refresh
End Sub
```

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

BeforeDelete Event

See Also Example Applies To

Occurs before a selected record is deleted in a **DataGrid** control.

Syntax

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

The BeforeDelete 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.
cancel	A Boolean expression that determines whether a record is deleted, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	Leaves focus on control and doesn't delete the record.
False	(Default) Continues with delete operation and enables change of focus.

Remarks

When the user selects a record selector in the control and presses DEL or CTL+X, the BeforeDelete event is triggered before the selected row is deleted.

Once the row is deleted, the AfterDelete event is triggered. The row selected for deletion is available in the collection provided by the **SelBookmarks** property.

If your event procedure sets the *cancel* argument to **True**, the row isn't deleted.

If more than one row is selected, the error message `Multiple rows cannot be deleted` is displayed.

© 2018 Microsoft

Visual Basic: DataGrid Control

BeforeDelete Event Example

This example displays a message that asks the user to confirm a deletion in a grid.

```
Private Sub DataGrid1_BeforeDelete (Cancel As Integer)
   Dim mResult As Integer
   mResult = MsgBox("Are you sure that you want to delete " & DataGrid1.SeletedRows &  " record?", _
    vbYesNo And vbQuestion, "Delete Confirmation")
   If mResult = vbNo Then Cancel = True
End Sub
```

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

BeforeInsert Event

See Also Example Applies To

Occurs before new records are inserted into a **DataGrid** control.

Syntax

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

The BeforeInsert 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.
cancel	A Boolean expression that determines if a record is added, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	Leaves focus on control and doesn't add a new record
False	(Default) Continues with copy and enables change of focus

Remarks

When the user selects the new record (at the bottom of the **DataGrid** control) and enters a character in one of the cells, the BeforeInsert event is triggered, followed by the BeforeUpdate, AfterUpdate and AfterInsert events.

If your event procedure sets the *cancel* argument to **True**, the row isn't inserted and the cell is cleared.

When the BeforeInsert event is triggered, the record has not been added to the database. The new record exists in the **DataGrid** control's copy buffer until this event procedure ends.

After the AfterInsert event is finished, the new record row in the **DataGrid** control is reinitialized and the edited record becomes the last row in the **DataGrid** control.

© 2018 Microsoft

Visual Basic: DataGrid Control

BeforeInsert Event Example

This example displays a message that asks the user to confirm the addition of a new record.

```
Private Sub DataGrid1_BeforeInsert (Cancel As Integer)
   Dim mResult As Integer
   mResult = MsgBox("Confirm: Add a new record?", _
    vbYesNo And vbQuestion, "Confirmation")
   If mResult = vbNo Then Cancel = True
End Sub
```

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

BeforeLabelEdit Event (ListView, TreeView Controls)

See Also Example Applies To

Occurs when a user attempts to edit the label of the currently selected **ListItem** or **Node** object.

Syntax

```
Private Sub object_BeforeLabelEdit(cancel As Integer)
```

The BeforeLabelEdit event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
cancel	An integer that determines if the operation is canceled. Any nonzero integer cancels the operation. The default is 0.

Remarks

Both the AfterLabelEdit and the BeforeLabelEdit events are generated only if the **LabelEdit** property is set to 0 (Automatic), or if the **StartLabelEdit** method is invoked.

The BeforeLabelEdit event occurs after the standard **Click** event.

To begin editing a label, the user must first click the object to select it, and click it a second time to begin the operation. The BeforeLabelEdit event occurs after the second click.

To determine which object's label is being edited, use the **SelectedItem** property. The following example checks the index of a selected **Node** before allowing an edit. If the index is 1, the operation is cancelled.

```
Private Sub TreeView1_BeforeLabelEdit(Cancel As Integer)
   If TreeView1.SelectedItem.Index = 1 Then
      Cancel = True    ' Cancel the operation
   End If
End Sub
```

© 2018 Microsoft

Visual Basic: Windows Controls

BeforeLabelEdit Event (ListView, TreeView Controls) Example

This example adds several **Node** objects to a **TreeView** control. If you try to edit a label, the **Node** object's index is checked. If it is 1, the edit is prevented. 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 try to edit the labels.

```
Private Sub Form_Load()
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,"P1","Parent 1")
   Set nodX = TreeView1.Nodes.Add("P1",tvwChild,,"Child 1")
   Set nodX = TreeView1.Nodes.Add("P1",tvwChild,,"Child 2")
   nodX.EnsureVisible   ' Make sure all nodes are visible.
End Sub

Private Sub TreeView1_BeforeLabelEdit(Cancel As Integer)
   ' Check selected node's index. If it is 1,
   ' then cancel the editing operation.
   If TreeView1.SelectedItem.Index = 1 Then
      MsgBox "Can't edit " + TreeView1.SelectedItem.Text
      Cancel = True
   End If
End Sub
```

This example adds several **ListItem** objects to a **ListView** control. If you try to edit a label, the **ListItem** object's index is checked. If it is 1, the edit is prevented. To try the example, place a **ListView** control on a form and paste the code into the form's Declarations section. Run the example, and try to edit the labels.

```
Private Sub Form_Load()
   Dim nodX As ListViewItem
   Set nodX = ListView1.ListItems.Add(, , "Item 1")
   Set nodX = ListView1.ListItems.Add(, , "Item 2")
   Set nodX = ListView1.ListItems.Add(, , "Item 3")
End Sub

Private Sub ListView1_BeforeLabelEdit(Cancel As Integer)
   ' Check selected item's index. If it is 1,
   ' then cancel the editing operation.
   If ListView1.SelectedItem.Index = 1 Then
      MsgBox "Can't edit " + ListView1.SelectedItem.Text
         Cancel = True
   End If
End Sub
```

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

BeforeLoadFile Event

See Also Example Applies To

Occurs when a component is added (not opened) to a project, or when a component's associated binary file (such as an .Frx file) is accessed.

Syntax

Sub *object*_**BeforeLoadFile(***vbproject* **As VBProject**, *filenames()* **As String)**

The BeforeLoadFile event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
vbproject	A **VBProject** object specifying the name of the project in which the file is to be loaded.
filenames	A string expression specifying the names of the files to be loaded.

Remarks

This event occurs in all add-ins that are connected to the **FileControl** object. This event occurs several times for a project: once for the project file; once for all the forms, modules, classes, **User** controls, **Property Pages**, and control files; and once for each of the .Frx files. This event occurs if a form file with an associated .Frx file is saved, because the .Frx is loaded when the .Frm file is saved.

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.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

BeforeUpdate Event

See Also Example Applies To

Occurs before data is moved from a **DataGrid** control to the control's copy buffer.

Syntax

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

The BeforeUpdate 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.
cancel	A Boolean expression that determines if data is copied, as described in Settings.

Settings

The settings for *cancel* are:

Setting	Description
True	Leaves focus on control and doesn't copy data.
False	(Default) Continues with copy operation and enables change of focus.

Remarks

When the user moves to another row or the **Recordset** object's **Update** method is executed, data is moved from the **DataGrid** control's copy buffer to the **Data** control's copy buffer and written to the database.

Just before the data is moved from the **DataGrid** control's copy buffer back into the **Data** control's copy buffer, the BeforeUpdate event is triggered. Unless the copy operation is canceled, the AfterUpdate event is triggered after the data has been moved back into the **Data** control's copy buffer and written to the database. The updated record is available by using the **Bookmark** property of the **DataGrid** control.

If you set the BeforeUpdate event *cancel* argument to **True**, focus remains on the control, neither the AfterUpdate or LostFocus event is triggered, and the record isn't saved to the database.

The BeforeUpdate event occurs before the AfterUpdate and LostFocus events for this control, or before the GotFocus event for the next control in the tab order.

This event occurs even if the control isn't bound.

Unlike the Change event, changing data in a control or record using code doesn't trigger this event.

You can use this event to validate data in a bound control record before permitting the user to commit the change to the **Data** control's copy buffer. By setting the *cancel* argument to **True**, the user can't move focus from the control until the application determines whether the data can be safely moved back to the **Data** control's copy buffer.

© 2018 Microsoft

Visual Basic: DataGrid Control

BeforeUpdate Event Example

This example displays a message that tells the user to enter a value in the first column before the grid can be updated.

```
Private Sub DataGrid1_BeforeUpdate (Cancel As Integer)
   If DataGrid1.Columns(1).Value = "" Then
      MsgBox "You must enter value in the first column!"
      Cancel = True
   End If
End Sub
```

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

BeginRequest Event

See Also Example Applies To

Occurs when the user selects an element on an HTML page that sends a request to the **WebClass** object. Marks the beginning of processing for an HTTP request.

Syntax

Private Sub *object*_**BeginRequest()**

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

Remark

This event is raised before any other events when an HTTP request is received.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: RDO Data Control

Visual Studio 6.0

BeginTrans Event

See Also Example Applies To

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

Syntax

Private Sub *object*.**BeginTrans**( )

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

Remarks

The BeginTrans event is raised after a **BeginTrans** method has completed. This event procedure can synchronize some other process with the transaction.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

ButtonClick Event (DataGrid Control)

See Also Example Applies To

Occurs when the current cell's built-in button is clicked.

Syntax

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

The ButtonClick 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 whose button was clicked.

Remarks

The built-in button is enabled for a column when its **Button** property is set to **True**.

Typically, you enable the column button when you want to drop down a **Visual Basic** control (such as the built-in combo box, a bound list box, or even another **DataGrid** control) for editing or data entry. When the button in the current cell is clicked, the ButtonClick event will be fired. You can then write code to drop down the desired control from the cell.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Multimedia MCI Control

Visual Studio 6.0

ButtonClick Event (Multimedia MCI Control)

See Also Example Applies To

Occurs when the user presses and releases the mouse button over one of the buttons in the **Multimedia MCI** control.

Syntax

Private Sub *MMControl_**Button***Click (*Cancel* **As Integer**)

Remarks

Button may be any of the following: Back, Eject, Next, Pause, Play, Prev, Record, Step, or Stop.

Each of the *Button*Click events, by default, perform an **MCI** command when the user chooses a button. The following table lists the **MCI** commands performed for each button in the control.

Button	Command
Back	MCI_STEP
Step	MCI_STEP
Play	MCI_PLAY
Pause	MCI_PAUSE
Prev	MCI_SEEK
Next	MCI_SEEK
Stop	MCI_STOP
Record	MCI_RECORD
Eject	MCI_SET with the MCI_SET_DOOR_OPEN parameter

Setting the *Cancel* parameter for the *Button*Click event to **True** prevents the default **MCI** command from being performed. The *Cancel* parameter can take either of the following settings.

Setting	Description

True	Prevents the default MCI command from being performed.
False	Performs the MCI command associated with the button after performing the body of the appropriate *Button*Click event.

The body of an event procedure is performed before performing the default **MCI** command associated with the event. Adding code to the body of the *Button*Click events augments the functionality of the buttons. If you set the *Cancel* parameter to **True** within the body of an event procedure or pass the value **True** as the argument to a *Button*Click event procedure, the default **MCI** command associated with the event will not be performed.

Note Issuing a **Pause** command to restart a paused device can end pending notifications from the original **Play** command if the device does not support the **MCI Resume** command. The **Multimedia MCI** control uses the **MCI Play** command to restart devices that do not support the **MCI Resume** command. Notifications from the **Play** command that restarts a paused device cancel callback conditions and supersede pending notifications from the original **Play** command.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

ButtonClick Event

See Also Example Applies To

Occurs when the user clicks on a **Button** object in a **Toolbar** control.

Syntax

Private Sub *object*_**ButtonClick**(**ByVal** *button* **As Button**)

The ButtonClick event syntax has these parts:

Part	Description
object	An object expression that evaluates to a **Toolbar** control.
button	A reference to the clicked **Button** object.

Remarks

To program an individual **Button** object's response to the ButtonClick event, use the value of the *button* argument. For example, the following code uses the **Key** property of the **Button** object to determine the appropriate action.

```
Private Sub Toolbar1_ButtonClick(ByVal Button As Button)
   Select Case Button.Key
   Case "Open"
      CommonDialog1.ShowOpen
   Case "Save"
      CommonDialog1.ShowSave
   End Select
End Sub
```

Note Because the user can rearrange **Button** objects using the Customize Toolbar dialog box, the value of the **Index** property may not always indicate the position of the button. Therefore, it's preferable to use the value of the **Key** property to retrieve a **Button** object.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Multimedia MCI Control

Visual Studio 6.0

ButtonCompleted Event (Multimedia MCI Control)

See Also Example Applies To

Occurs when the **MCI** command activated by a **Multimedia MCI** control button finishes.

Syntax

Private Sub *MMControl_**Button***Completed** (*Errorcode* **As Long**)

Remarks

Button may be any of the following: Back, Eject, Next, Pause, Play, Prev, Record, Step, or Stop.

The *Errorcode* argument can take the following settings.

Setting	Description
0	Command completed successfully.
Any other value	Command did not complete successfully.

If the *Cancel* argument is set to **True** during a *Button*Click event, the *Button*Completed event is not triggered.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

ButtonDropDown Event

See Also Example Applies To

Occurs when the user clicks the dropdown arrow on a **Button** object.

Syntax

Private Sub *object*_**ButtonDropDown(ByVal** *Button* **As ComctlLib.Button)**

The ButtonDropDown event syntax has these parts:

Part	Description
object	Required.
Button	Returns a reference to the clicked **Button** object.

Remarks

The dropdown arrow only appears when a **Button** object's **Style** is set to **tbrDropdown**.

The ButtonDropDown event occurs before the ButtonMenuClick event. Use the ButtonDropDown event to determine what items exist in the **ButtonMenus** collection, and edit them as needed.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Multimedia MCI Control

Visual Studio 6.0

ButtonGotFocus Event (Multimedia MCI Control)

See Also Example Applies To

Occurs when a button in the **Multimedia MCI** control receives the input focus.

Syntax

Private Sub *MMControl_**Button***GotFocus ()

Remarks

Button may be any of the following: Back, Eject, Next, Pause, Play, Prev, Record, Step, or Stop.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Multimedia MCI Control

Visual Studio 6.0

ButtonLostFocus Event (Multimedia MCI Control)

See Also Example Applies To

Occurs when a button in the **Multimedia MCI** control loses the input focus.

Syntax

Private Sub *MMControl_**Button***LostFocus ()

Remarks

Button may be any of the following: Back, Eject, Next, Pause, Play, Prev, Record, Step, or Stop.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

ButtonMenuClick Event

See Also Example Applies To

Occurs when the user clicks a **ButtonMenu** object.

Syntax

Private Sub *object*_**ButtonMenuClick**([*index* **As Integer**,]**ByVal** *ButtonMenu* **As ComctlLib.ButtonMenu**)

The ButtonMenuClick 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.
ButtonMenu	A reference to the clicked **ButtonMenu** object.

Remarks

You can use the ButtonMenuClick event with the **ButtonMenu** object's **Parent** property to determine which button was clicked.

© 2018 Microsoft

Visual Basic: Windows Controls

ButtonMenu Object, ButtonMenuClick Event Example

The example adds five **Button** objects to a **Toolbar** control and also adds two **ButtonMenu** objects to each **Button** object. When a ButtonMenu object is clicked, the ButtonMenuClick event is used to determine its behavior. To try the example, place a Toolbar control on a form and paste the code into the Declarations section of the code module.

```
Option Explicit

Private Sub Form_Load()
   Dim i As Integer
   Dim btn As Button

   ' Add five Button objects to the Toolbar control.
   For i = 1 To 5
      Set btn = Toolbar1.Buttons.Add(Caption:= i, Style:= tbrDropDown)
      ' Add two ButtonMenu objects to each Button.
         btn.ButtonMenus.Add Text:="Help"
         btn.ButtonMenus.Add Text:="Options"
   Next i
End Sub

Private Sub Toolbar1_ButtonMenuClick(ByVal ButtonMenu As ComctlLib.ButtonMenu)
   Select Case ButtonMenu.Index
   Case 1
      MsgBox "Press the button."
   Case 2
      MsgBox "Offer some option"
   End Select
End Sub
```

© 2018 Microsoft