# Visual Basic Reference

**Visual Studio 6.0**

# Activate, Deactivate Events

See Also Example Applies To

- Activate occurs when an object becomes the active window.
- Deactivate occurs when an object is no longer the active window.

**Syntax**

```
Private Sub object_Activate( )

Private Sub object_Deactivate( )
```

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

**Remarks**

An object can become active by user action, such as clicking it, or by using the **Show** or **SetFocus** methods in code.

The Activate event can occur only when an object is visible. For example, a form loaded with the **Load** statement isn't visible unless you use the **Show** method or set the form's **Visible** property to **True**.

The Activate and Deactivate events occur only when moving the focus within an application. Moving the focus to or from an object in another application doesn't trigger either event. The Deactivate event doesn't occur when unloading an object.

The Activate event occurs before the GotFocus event; the LostFocus event occurs before the Deactivate event.

These events occur for MDI child forms only when the focus changes from one child form to another. In an **MDIForm** object with two child forms, for example, the child forms receive these events when the focus moves between them. However, when the focus changes between a child form and a non-MDI child form, the parent **MDIForm** receives the Activate and Deactivate events.

If an .exe file built by Visual Basic displays a dialog box created by a .dll file also built in Visual Basic, the .exe file's form will get the Deactivate and LostFocus events. This may be unexpected, because you should not get the Deactivate event:

- If the object is an out-of-process component.
- If the object isn't written in Visual Basic.
- In the development environment when calling a DLL built in Visual Basic.

# Visual Basic Reference

# Activate, Deactivate Events Example

This example updates the status bar text to display the caption of the active form. To try this example, create a **Form** object (Form1) and a new **MDIForm** object (MDIForm1). On MDIForm1, draw a **PictureBox** control containing a **Label** control. On Form1, set the **MDIChild** property to **True**. Paste the MDIForm_Load event procedure code into the Declarations section of the **MDIForm** object. Paste the Form_Activate event procedure code into the Declarations section of the MDI child form, and then press F5.

```
Private Sub MDIForm_Load ()
   Form1.Caption = "Form #1"       ' Set caption of Form1.
   Dim NewForm As New Form1      ' Create a new child form.
   Load NewForm
   NewForm.Caption = "Form #2"    ' Set caption of new form.
   NewForm.Show                  ' Display the new form.
End Sub

Private Sub Form_Activate ()
      ' Set status bar text.
   MDIForm1.Label1.Caption = "Current form: " & Me.Caption
End Sub
```

# Visual Basic: DataRepeater Control

**Visual Studio 6.0**

# ActiveRowChanged Event

See Also Example Applies To

Occurs when the **ActiveRow** property is set.

**Syntax**

*object*_**ActiveRowChanged**( )

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

# Visual Basic Extensibility Reference

**Visual Studio 6.0**

# AfterAddFile Event

See Also Example Applies To

Occurs after a component is added to the current Visual Basic project with the **Add File** command in the **Project** menu.

**Syntax**

**Sub** *object*_**AfterAddFile(***vbproject* **As VBProject**, *filetype* **As vbext_FileType**, *filename* **As String)**

The AfterAddFile 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 was added. |
| *filetype* | An enumerated value (**vbext_FileType**) specifying the type of file that was added, as listed in Settings. |
| *filename* | A string expression specifying the name of the file that was added. |

**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. |

### Remarks

Visual Basic triggers this event only for files you can add from the **Project** menu. (That is, forms, classes, **User** controls, **Property Pages**, and modules). The AfterAddFile event does not occur if you select **Add** *object* from the **Project** menu. It also does not occur when an .Frx file is created for the first time, and doesn't occur twice when a form is added.

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.
- Backup the file.

# Visual Basic Extensibility Reference

**Visual Studio 6.0**

# AfterCloseFile Event

See Also Example Applies To

Occurs after a project has been closed, either directly by the user, or by Visual Basic when the user quits the program.

**Syntax**

**Sub** *object*_**AfterCloseFile(***vbproject* **As VBProject**, *filetype* **As vbext_FileType**, *filename* **As String,** *wasdirty* **As Boolean)**

The AfterCloseFile 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 was closed. |
| *filetype* | An enumerated value (**vbext_FileType**) specifying the type of file that was closed, as listed in Settings. |
| *filename* | A string expression specifying the name of the file that was closed. |
| *wasdirty* | A Boolean expression that specifies whether changes were saved to a file prior to it being closed, as listed 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 *wasdirty* are:

| Setting | Description |
|---|---|
| **True** | The file was dirty when it was closed. (That is, the user elected to not save changes made to the file prior to closing it.) |
| **False** | The file was not dirty when it was closed (That is, the user selected to save changes made to the file prior to closing it.) |

**Remarks**

This event can occur once for each add-in connected to the **FileControl** object in each project; once for each form, module, class, and control file, and once for the project file.

The AfterCloseFile event does not occur if the form is dirty and the user selects **No** on the **Save changes to the following files** dialog box. Also, this event does not occur for .Frx files when a project is closed. It occurs 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.
- Compare versions of the executable (.EXE) file.

# Visual Basic: DataGrid Control

**Visual Studio 6.0**

# AfterColEdit Event

See Also Example Applies To

Occurs after editing is completed in a grid cell.

**Syntax**

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

The AfterColEdit 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 that was edited. |

**Remarks**

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, and data from the cell is moved to the grid's copy buffer. The AfterColEdit event immediately follows the AfterColUpdate event.

When editing is completed in a grid cell, this event is always triggered, even if no changes were made to the cell or the BeforeColUpdate event was canceled.

The AfterColEdit event will not be fired if the BeforeColEdit event is canceled.

# Visual Basic: DataGrid Control

**Visual Studio 6.0**

# AfterColUpdate Event

See Also Example Applies To

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

**Syntax**

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

The AfterColUpdate 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 in the control. |

**Remarks**

When a user completes editing within a **DataGrid** control cell, as when tabbing to another column in the same row, pressing ENTER, or when the control loses focus, the BeforeColUpdate event is executed, and unless canceled, data from the cell is moved to the control's copy buffer. Once moved, the AfterColUpdate event is executed.

The AfterColUpdate event occurs after the BeforeColUpdate event, and only if the *cancel* argument in the BeforeColUpdate event is not set to **True**.

Once the AfterColUpdate event procedure begins, the cell data has already been moved to the control's copy buffer and can't be canceled, but other updates can occur before the data is committed to the **Recordset**.

# Visual Basic: DataGrid Control

# AfterColUpdate Event Example

This example does a lookup when one column is updated and places the result in another column.

```
Private Sub DataGrid1_AfterColUpdate (ColIndex As Integer)
   If ColIndex = 1 Then
      Data1.Recordset.FindFirst "PubId = " _
       & DataGrid1.Columns(1).Value
      If Not Data1.Recordset.NoMatch Then
         DataGrid1.Columns(2).Value = _
          Data1.Recordset.Fields("Publisher")
      Else
         DataGrid1.Columns(2).Value = "No Match"
      End If
   End If
End Sub
```

# Visual Basic: DataGrid Control

**Visual Studio 6.0**

# AfterDelete Event

See Also Example Applies To

Occurs after the user deletes a selected record in the **DataGrid** control.

**Syntax**

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

The AfterDelete 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. |

**Remarks**

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

# Visual Basic: DataGrid Control

# AfterDelete Event Example

This example displays a message confirming that a record has successfully been deleted.

```
Private Sub DataGrid1_AfterDelete ()
   MsgBox "Record has successfully been deleted!"
End Sub
```

# Visual Basic Extensibility Reference

**Visual Studio 6.0**

# AfterChangeFileName Event

See Also Example Applies To

Occurs after a file in the current project has been saved for the first time, or saved with a new name. It also occurs when the project is first compiled to an .Exe file, or when compiled to a new .Exe name.

**Syntax**

**Sub** *object*_**AfterChangeFileName (***vbproject* **As VBProject**, *filetype* **As vbext_FileType**, *newname* **As String**, *oldname* **As String)**

The AfterChangeFileName 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 was changed. |
| *filetype* | An enumerated value (**vbext_FileType**) specifying the type of file that was changed, as listed in Settings. |
| *newname* | A string expression specifying the new name of the file. |
| *oldname* | A string expression specifying the old name of the file. |

**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. |

# Visual Basic: DataGrid Control

**Visual Studio 6.0**

# AfterInsert Event

See Also Example Applies To

Occurs after the user inserts a new record into the **DataGrid** control.

**Syntax**

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

The AfterInsert 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. |

**Remarks**

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

When the AfterInsert event is triggered, the record has already been added to the database. The **Bookmark** property can be used to access the new record.

The AfterInsert event can't be canceled.

The AfterInsert event procedure can be used to update other tables or to perform post-update cleanup of other controls.

# Visual Basic: DataGrid Control

# AfterInsert Event Example

This example creates an entry in a related table if the user enters a value in a column in the grid.

```
Private SubDataGrid1_AfterInsert ()
   If DataGrid1.Columns(1).Value <> "" Then
      Data2.Recordset.AddNew
      Data2.Recordset.Fields("PubId") = DataGrid1.Columns(1).Value
      Data2.Recordset.Update
   End If
End Sub
```

# Visual Basic: Windows Controls

**Visual Studio 6.0**

# AfterLabelEdit Event (ListView, TreeView Controls)

See Also Example Applies To

Occurs after a user edits the label of the currently selected **Node** or **ListItem** object.

**Syntax**

```
Private Sub object_AfterLabelEdit(cancel As Integer, newstring As String)
```

The AfterLabelEdit 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 label editing operation is canceled. Any nonzero integer cancels the operation. Boolean values are also accepted. |
| *newstring* | The string the user entered, or **Null** if the user canceled the operation. |

**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 AfterLabelEdit event is generated after the user finishes the editing operation, which occurs when the user clicks on another **Node** or **ListItem** or presses the ENTER key.

To cancel a label editing operation, set *cancel* to any nonzero number or to **True**. If a label editing operation is canceled, the previously existing label is restored.

The *newstring* argument can be used to test for a condition before canceling an operation. For example, the following code cancels the operation if *newstring* is a number:

```
Private Sub TreeView1_AfterLabelEdit(Cancel As Integer, NewString As String)
   If IsNumeric(NewString) Then
      MsgBox "No numbers allowed"
      Cancel = True
   End If
End Sub
```

# Visual Basic: Windows Controls

# AfterLabelEdit Event (ListView, TreeView Controls) Example

This example adds three **Node** objects to a **TreeView** control. When you attempt to edit a **Node** object's label, the object's index is checked. If it is 1, the operation is canceled. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, click twice on the top **Node** object's label to edit it, type in some text, and press ENTER.

```
Private Sub Form_Load()
   TreeView1.Style = tvwTreelinesText ' Lines and text.
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(,,,"Parent")
   Set nodX = TreeView1.Nodes.Add(1,tvwChild,,"Child1")
   Set nodX = TreeView1.Nodes.Add(1,tvwChild,,"Child2")
   nodX.EnsureVisible   ' Make sure all nodes are visible.
End Sub

Private Sub TreeView1_AfterLabelEdit _
(Cancel As Integer, NewString As String)
   ' If current node's index is 1, edit is canceled.
   If TreeView1.SelectedItem.Index = 1 Then
      Cancel = True
      MsgBox "Can't replace " & TreeView1.SelectedItem.Text & _
      " with " & NewString
   End If
End Sub
```

This example adds three **ListItem** objects to a **ListView** control. When you attempt to edit a **ListItem** object's label, the object's index is checked. If it is 1, the operation is canceled. To try the example, place a **ListView** control on a form and paste the code into the form's Declarations section. Run the example, click twice on any **ListItem** object's label to edit it, type in some text, and press ENTER.

```
Private Sub Form_Load()
   Dim itmX As ListItem
   Set itmX = ListView1.ListItems.Add(,,"Item1")
   Set itmX = ListView1.ListItems.Add(,,"Item 2")
   Set itmX = ListView1.ListItems.Add(,,"Item 3")
End Sub

Private Sub ListView1_AfterLabelEdit _
(Cancel As Integer, NewString As String)
   ' If current ListItem's index is 1, edit is canceled.
   If ListView1.SelectedItem.Index = 1 Then
      Cancel = True
      MsgBox "Can't replace " & ListView1.SelectedItem.Text & _
      " with " & NewString
   End If
End Sub
```

# Visual Basic Extensibility Reference

**Visual Studio 6.0**

# AfterRemoveFile Event

See Also Example Applies To

Occurs after a file is removed from the active Visual Basic project.

**Syntax**

**Sub** *object*_**AfterRemoveFile(***vbproject* **As VBProject**, *filetype* **As vbext_FileType**, *filename* **As String)**

The AfterRemoveFile 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 from which the file was removed. |
| *filetype* | An enumerated value (**vbext_FileType**) specifying the type of file that was removed, as listed in Settings. |
| *filename* | A string expression specifying the name of the file that was removed. |

**Remarks**

The AfterRemoveFile event does not occur for components that are removed before they have been 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.

# Visual Basic: DataGrid Control

**Visual Studio 6.0**

# AfterUpdate Event

See Also Example Applies To

Occurs after changed data has been written to the database from a **DataGrid** control.

**Syntax**

**Sub** *object*_**AfterUpdate (***index* **As Integer)**

The AfterUpdate 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. |

**Remarks**

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

The updated record is available by using the **Bookmark** property of the **DataGrid** control.

The AfterUpdate event occurs after the BeforeUpdate event, but before the LostFocus event for the control (or GotFocus for the next control in the tab order). This event occurs in bound and unbound mode and can't be canceled.

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

# Visual Basic: DataGrid Control

# AfterUpdate Event Example

This example updates a label when any change has been made in the grid.

```
Private Sub DataGrid1_AfterUpdate ()
   Label1.Caption = "Last modified: " & Format$(Now, "Long Date")
End Sub
```

# Visual Basic Extensibility Reference

**Visual Studio 6.0**

# AfterWriteFile Event

See Also Example Applies To

Occurs after a file is written to disk.

**Syntax**

**Sub** *object*_**AfterWriteFile(***vbproject* **As VBProject**, *filetype* **As vbext_FileType**, *filename* **As String,** *result* **As Integer)**

The AfterWriteFile 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 to which the file was written. |
| *filetype* | An enumerated value (**vbext_FileType**) specifying the type of file that was written, as listed in Settings. |
| *filename* | A string expression specifying the name of the file that was written. |
| *result* | A numeric expression that specifies the result of the write operation, as listed 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 *result* are:

| Value | Description |
|---|---|
| 0 | Write was successful. |
| 1 | Write was canceled. |
| 2 | Write failed. |

**Remarks**

The AfterWriteFile event occurs when the binary data file associated with a component (such as an .Frx file) is saved for the first time, and 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.
- Compare versions of the executable (.EXE) file.

# Visual Basic Reference

**Visual Studio 6.0**

# AmbientChanged Event

See Also Example Applies To

Occurs when an ambient propertys value changes.

**Syntax**

**Sub** *object*_**AmbientChanged(***PropertyName* **As String)**

The AmbientChanged event syntax has these parts:

| Part | Description |
|---|---|
| *object* | An object expression that evaluates to an object in the Applies To list. |
| *PropertyName* | A string that identifies the ambient property that has changed. |

**Remarks**

Using *PropertyName*, the control can access the **AmbientProperties** object in the **Ambient** property to check for the new value of the changed ambient property.

If an instance of the control is placed on a Visual Basic form, and the **FontTransparent** property of the form is changed, the **AmbientChanged** event will not be raised.

# Visual Basic Reference

**Visual Studio 6.0**

# ApplyChanges Event

See Also Example Applies To

Occurs when the user presses the **OK** button or the **Apply** button on the property page, or when property pages are switched by selecting tabs.

**Syntax**

**Sub** *object*_**ApplyChanges()**

The ApplyChanges event syntax has these parts:

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

**Remarks**

When the ApplyChanges event is raised, the author of the property page needs to handle the setting of all the new property values to the controls; hopefully the author kept track of which properties were changed, otherwise all properties will have to be set. To know what controls are to be changed, use the **SelectedControls** property.

The ApplyChanges event will be raised only if the **Changed** property is set to **True**.

# Visual Basic: RDO Data Control

**Visual Studio 6.0**

# Associate Event

See Also Example Applies To

Fired after a new connection is associated with the object.

**Syntax**

**Private Sub** *object*.**Associate**( )

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

**Remarks**

This event is raised after the result set is associated with a new **rdoConnection** object. You can use this event to initialize the new connection. The **ActiveConnection** property of the associated **rdoResultset** object refers to the new connection.

For example, you can use the Associate event procedure to send a special query each time a connection is established, but before other operations are executed.

# Visual Basic Reference

**Visual Studio 6.0**

# AsyncProgress Event

See Also Example Applies To

Occurs during an asynchronous operation, as each page of data has been processed by the specific operation (determined by the *JobType* argument).

**Syntax**

**Private Sub** *object*_**AsyncProgress(***JobType* **As AsyncTypeConstants**, *Cookie* **As Long**, *PageCompleted* **As Long**, *TotalPages* **As Long)**

The AsyncProgress event syntax has these parts:

| Part | Description |
|---|---|
| *object* | Required. An object expression that evaluates to an object in the Applies To list. |
| *JobType* | An integer that specifies the type of operation, as shown in Settings. |
| *Cookie* | A Long integer that identifies the specific operation. |
| *PageCompleted* | A variant that returns the number of completed pages. |
| *TotalPages* | A variant that returns the total number of pages to be processed. |

**Settings**

The *JobType* settings are:

| Constant | Value | Description |
|---|---|---|
| **rptAsyncPreview** | 0 | Not applicable for the **AsyncProgress** event. |
| **rptAsyncPrint** | 1 | The report is processing a PrintReport operation. |
| **rptAsyncReport** | 2 | The report is processing an ExportReport operation. |

**Note** The constant for preview operations will not occur with the **AsyncProgress** event. However, the constant can be used for the ProcessingTimeOut event.

# Visual Basic Reference

**Visual Studio 6.0**

# AsyncReadComplete Event

See Also Example Applies To

Occurs when the container has completed an asynchronous read request.

**Syntax**

**Sub** *object*_**AsyncReadComplete(***AsyncProp* **As AsyncProperty)**

The AsyncReadComplete event syntax has these parts:

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

**Remarks**

The value in *AsyncProp* specifies the particular asynchronous data read request that has completed, and matches the value given in a previous **AsyncRead** method invocation.

Error handling code should be placed in the AsyncReadComplete event procedure, because an error condition may have stopped the download. If this was the case, that error will be raised when the Value property of the **AsyncProperty** object is accessed.

# Visual Basic Reference

**Visual Studio 6.0**

# AsyncReadProgress Event

See Also Example Applies To

Occurs when more data is available as a result of an **AsyncRead** method.

**Syntax**

**Sub** *object*_**AsyncReadProgress(***AsyncProp* **As AsyncProperty)**

The AsyncReadProgress event syntax has these parts:

| Part | Description |
|---|---|
| *Object* | An object expression that evaluates to an object in the Applies To list. |
| *AsyncProp* | An **AsyncProperty** object. |

**Remarks**

The AsyncReadProgress event gives you access to the progressive status of an asynchronous download.

If the **AsyncType** for the **AsyncRead** method is set to either **vbAsyncTypeFileName** or **vbAsyncTypeByteArray**, you can have access to the partially downloaded data during a download. Normally, Visual Basic keeps the file open during the entire download process. Getting the **Value** for **vbAsyncTypeFileName** slows the download process (but not significantly over a slow link), because Visual Basic must close the file each time before firing the event, and then re-open the file the next time data is available.

If you open the file, then you must close it before the code inside the AsyncReadProgress event completes (or before calling **DoEvents** or otherwise allowing a Windows message to be dispatched), otherwise an error will occur, preventing the AsyncReadProgress event from firing. To avoid this, include error-handling code in the AsyncReadComplete event procedure.

**Note** The firing of the AsyncReadProgress and AsyncReadComplete events is message-based. That is, a Windows notification message is posted and must be dispatched by the applications message pump to fire. Therefore, you should avoid loop structures in the code of these events. Also avoid using **DoEvents**, since it'll unexpectedly cause the code to be re-entrant.

Both the AsyncReadProgress and AsyncReadComplete events occur when the download completes. (That is, **AsyncProp.StatusCode** = **vbAsyncStatusCodeEndDownloadData**.)

**Visual Studio 6.0**

*Visual Basic: MSChart Control*

# AxisActivated Event

See Also Example Applies To

Occurs when the user double clicks on a chart axis.

**Syntax**

**Private Sub** *object*_**AxisActivated** (*axisId* **As Integer**, *axisIndex* **As Integer**, *mouseFlags* **As Integer**, *cancel* **As Integer** )

The AxisActivated event syntax has these parts:

| Part | Description |
|---|---|
| *object* | An object expression that evaluates to an object in the Applies To list. |
| *axisId* | Integer. An integer that identifies a specific axis, as described in Settings. |
| *axisIndex* | Integer. An integer reserved for future use. For this version of MSChart control, 1 is the only valid value for this argument. |
| *mouseFlags* | Integer. An integer that indicates whether a key is held down when the mouse button is clicked, as described in Settings. |
| *cancel* | Integer. An integer that is not used at this time. |

**Settings**

The event handler determines which axis is activated and sets *axisId* to:

| Constants | Description |
|---|---|
| **VtChAxisIdX** | If the *x* axis is affected. |
| **VtChAxisIdY** | If the *y* axis is affected. |
| **VtChAxisIdY2** | If the secondary *y* axis is affected. |
| **VtChAxisIdZ** | If the *z* axis is affected. |

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*

# AxisLabelActivated Event

See Also Example Applies To

Occurs when the user double clicks on an axis label.

**Syntax**

**Private Sub** *object*_**AxisLabelActivated** (*axisId* **As Integer**, *axisIndex* **As Integer**, *labelSetIndex* **As Integer**, *labelIndex* **As Integer**, *mouseFlags* **As Integer**, *cancel* **As Integer**)

The AxisLabelActivated event syntax has these parts:

| Part | Description |
|---|---|
| *object* | An object expression that evaluates to an object in the Applies To list. |
| *axisId* | Integer. An integer that identifies a specific axis, as described in Settings. |
| *axisIndex* | Integer. An integer reserved for future use. For this version of MSChart control, 1 is the only valid value for this argument. |
| *labelSetIndex* | Integer. An integer that identifies the level of labels you are double clicking on. Levels of labels are numbered from the axis out, beginning with 1. |
| *labelIndex* | Integer. An integer that is currently unused. |
| *mouseFlags* | Integer. An integer that indicates if a key is held down when the mouse button is clicked. |
| *cancel* | Integer. An integer that is not used at this time. |

**Settings**

The event handler determines which axis label is activated and sets *axisId* to:

| Constants | Description |
|---|---|
| **VtChAxisIdX** | If the *x* axis is affected. |
| **VtChAxisIdY** | If the *y* axis is affected. |

| | |
|---|---|
| **VtChAxisIdY2** | If the secondary *y* axis is affected. |
| **VtChAxisIdZ** | If the *z* axis is affected. |

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*

# AxisLabelSelected Event

See Also Example Applies To

Occurs when the user clicks an axis label.

**Syntax**

**Private Sub** *object*_**AxisLabelSelected** ( *axisId* **As Integer**, *axisIndex* **As Integer**, *labelSetIndex* **As Integer**, *labelIndex* **As Integer**, *mouseFlags* **As Integer**, *cancel* **As Integer** )

The AxisLabelSelected event syntax has these parts.

| Part | Description |
|---|---|
| *object* | An object expression that evaluates to an object in the Applies To list. |
| *axisId* | Integer. An integer that identifies a specific axis, as described in Settings. |
| *axisIndex* | Integer. An integer reserved for future use. For this version of MSChart control, 1 is the only valid value for this argument. |
| *labelSetIndex* | Integer. An integer that identifies the level of labels you are double clicking on. Levels of labels are numbered from the axis out, beginning with 1. |
| *labelIndex* | Integer. An integer that is currently unused. |
| *mouseFlags* | Integer. An integer that indicates if a key is held down when the mouse button is clicked, as described in Settings. |
| *cancel* | Integer. An integer that is not used at this time. |

**Settings**

The event handler determines which axis label is selected and sets *axisId* to:

| Constants | Description |
|---|---|
| **VtChAxisIdX** | If the *x* axis is affected. |
| **VtChAxisIdY** | If the *y* axis is affected. |

| | |
|---|---|
| **VtChAxisIdY2** | If the secondary *y* axis is affected. |
| **VtChAxisIdZ** | If the *z* axis is affected. |

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*

# AxisLabelUpdated Event

See Also Example Applies To

Occurs when an axis label has changed.

**Syntax**

**Private Sub** *object*_**AxisLabelUpdated** (*axisId* **As Integer**, *axisIndex* **As Integer**, *labelSetIndex* **As Integer**, *labelIndex* **As Integer**, *updateFlags* **As Integer**)

The AxisLabelUpdated event syntax has these parts:

| Part | Description |
|---|---|
| *object* | An object expression that evaluates to an object in the Applies To list. |
| *axisId* | Integer. An integer that identifies a specific axis, as described in Settings. |
| *axisIndex* | Integer. An integer reserved for future use. For this version of MSChart control, 1 is the only valid value for this argument. |
| *labelSetIndex* | Integer. An integer that identifies the level of labels you are double clicking on. Levels of labels are numbered from the axis out, beginning with 1. |
| *labelIndex* | Integer. An integer that is currently unused. |
| *updateFlags* | Integer. An integer provides information about the update of the label, as described in Settings. |

**Settings**

The event handler determines which axis label is updated and sets *axisId* to:

| Constants | Description |
|---|---|
| **VtChAxisIdX** | If the *x* axis is affected. |
| **VtChAxisIdY** | If the *y* axis is affected. |
| **VtChAxisIdY2** | If the secondary *y* axis is affected. |

| | |
|---|---|
| **VtChAxisIdZ** | If the *z* axis is affected. |

The event handler determines the affect of the update, and sets *updateFlags* to:

| Constants | 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. |

*Visual Basic: MSChart Control*

# AxisSelected Event

See Also Example Applies To

Occurs when the user clicks on a chart axis.

**Syntax**

**Private Sub** *object*_**AxisSelected** (*axisId* **As Integer**, *axisIndex* **As Integer**, *mouseFlags* **As Integer**, *cancel* **As Integer** )

The AxisSelected event syntax has these parts:

| Part | Description |
|---|---|
| *object* | An object expression that evaluates to an object in the Applies To list. |
| *axisId* | Integer. An integer that identifies a specific axis, as described in Settings. |
| *axisIndex* | Integer. An integer reserved for future use. For this version of MSChart control, 1 is the only valid value for this argument. |
| *mouseFlags* | Integer. An integer that indicates whether key is held down when the mouse button is clicked, as described in Settings. |
| *cancel* | Integer. An integer that is not used at this time. |

**Settings**

The event handler determines which axis is selected and sets *axisId* to:

| Constants | Value | Description |
|---|---|---|
| **VtChAxisIdX** | 0 | If the *x* axis is affected. |
| **VtChAxisIdY** | 1 | If the *y* axis is affected. |
| **VtChAxisIdY2** | 2 | If the secondary *y* axis is affected. |
| **VtChAxisIdZ** | 3 | If the *z* axis is affected. |
| **VtChAxisIDNone** | 4 | No axis. |

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*

# AxisTitleActivated Event

See Also Example Applies To

Occurs when the user double clicks on an axis title.

**Syntax**

**Private Sub** *object*_**AxisTitleActivated** (*axisId* **As Integer**, *axisIndex* **As Integer**, *mouseFlags* **As Integer**, *cancel* **As Integer** )

The AxisTitleActivated event syntax has these parts:

| Part | Description |
|---|---|
| *object* | An object expression that evaluates to an object in the Applies To list. |
| *axisId* | Integer. An integer that identifies a specific axis. |
| *AxisIndex* | Integer. An integer reserved for future use. For this version of MSChart control, 1 is the only valid value for this argument. |
| *mouseFlags* | Integer. An integer that indicates whether a key is held down when the mouse button is clicked. |
| *cancel* | Integer. This argument is not used at this time. |

**Settings**

The event handler determines which axis title is activated and sets *axisId* to:

| Constants | Value | Description |
|---|---|---|
| **VtChAxisIdX** | 0 | If the *x* axis is affected. |
| **VtChAxisIdY** | 1 | If the *y* axis is affected. |
| **VtChAxisIdY2** | 2 | If the secondary *y* axis is affected. |
| **VtChAxisIdZ** | 3 | If the *z* axis is affected. |
| **VtChAxisIDNone** | 4 | No axis. |

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*

# AxisTitleSelected Event

See Also Example Applies To

Occurs when the user clicks on an axis title.

**Syntax**

**Private Sub** *object*_**AxisTitleSelected** ( *axisId* **As Integer**, *axisIndex* **As Integer**, *mouseFlags* **As Integer**, *cancel* **As Integer** )

The AxisTitleSelected event syntax has these parts:

| Part | Description |
|---|---|
| *object* | An object expression that evaluates to an object in the Applies To list. |
| *axisId* | Integer. An integer that identifies a specific axis. |
| *AxisIndex* | Integer. An integer reserved for future use. For this version of MSChart control, 1 is the only valid value for this argument. |
| *mouseFlags* | Integer. An integer that indicates whether a key is held down when the mouse button is clicked. |
| *cancel* | Integer. This argument is not used at this time. |

**Settings**

The event handler determines which axis title is selected and sets *axisId* to:

| Constants | Value | Description |
|---|---|---|
| **VtChAxisIdX** | 0 | If the *x* axis is affected. |
| **VtChAxisIdY** | 1 | If the *y* axis is affected. |
| **VtChAxisIdY2** | 2 | If the secondary *y* axis is affected. |
| **VtChAxisIdZ** | 3 | If the *z* axis is affected. |
| **VtChAxisIDNone** | 4 | No axis. |

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*

# AxisTitleUpdated Event

See Also Example Applies To

Occurs when an axis title has changed.

**Syntax**

**Private Sub** *object*_**AxisTitleUpdated** ( *axisId* **As Integer**, *axisIndex* **As Integer**, *updateFlags* **As Integer**)

The AxisTitleUpdated event syntax has these parts:

| Part | Description |
|---|---|
| *object* | An object expression that evaluates to an object in the Applies To list. |
| *axisId* | Integer. An integer that identifies a specific axis. |
| *axisIndex* | Integer. An integer reserved for future use. For this version of MSChart control, 1 is the only valid value for this argument. |
| *updateFlags* | Integer. An integer that provides information about the update of the title. |

**Settings**

The event handler determines which axis title is updated and sets *axisId* to:

| Constants | Value | Description |
|---|---|---|
| **VtChAxisIdX** | 0 | If the *x* axis is affected. |
| **VtChAxisIdY** | 1 | If the *y* axis is affected. |
| **VtChAxisIdY2** | 2 | If the secondary *y* axis is affected. |
| **VtChAxisIdZ** | 3 | If the *z* axis is affected. |
| **VtChAxisIDNone** | 4 | No axis. |

The following table lists the constants for *updateFlags*.

| Constants | 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. |

**Visual Studio 6.0**

*Visual Basic: MSChart Control*

# AxisUpdatedEvent

See Also Example Applies To

Occurs when an axis has changed.

**Syntax**

**Private Sub** *object*_**AxisUpdated** ( *axisId* **As Integer**, *axisIndex* **As Integer**, *updateFlags* **As Integer**)

The AxisUpdated event syntax has these parts:

| Part | Description |
|---|---|
| *object* | An object expression that evaluates to an object in the Applies To list. |
| *axisId* | Integer. An integer that identifies a specific axis. |
| *axisIndex* | Integer. An integer reserved for future use. For this version of MSChart control, 1 is the only valid value for this argument. |
| *updateFlags* | Integer. An integer that provides information about the update of the axis. |

**Settings**

The event handler determines which axis is updated and sets *axisId* to:

| Constants | Value | Description |
|---|---|---|
| **VtChAxisIdX** | 0 | If the *x* axis is affected. |
| **VtChAxisIdY** | 1 | If the *y* axis is affected. |
| **VtChAxisIdY2** | 2 | If the secondary *y* axis is affected. |
| **VtChAxisIdZ** | 3 | If the *z* axis is affected. |
| **VtChAxisIDNone** | 4 | No axis. |

The following table lists the constants for *updateFlags*.

| Constants | 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. |