# Event-Driven Programming Using IActiveViewEvents

*By Stephen Lead, ESRI (UK) Ltd.*

Programming in ArcGIS is essentially event driven. An event causes a process to run. User-driven events include pressing a button on a toolbar, zooming in on the display, or adding a scale bar to a layout. In many cases when the user interacts with ArcMap, events are also triggered automatically by background ArcGIS processes. For example, when the user adds a new layer to a map, ArcMap automatically refreshes the active display and fires events to show that an item has been added to the map and drawn to the display. By anticipating these events and trapping them, the developer can use these event triggers to customize the behavior of ArcMap for a particular purpose.

This article explains how the IActiveViewEvents interface works and how it can be used to program in ArcMap. If the ArcObjects Developer Kit was not originally installed with ArcGIS, install it before beginning this exercise. Basic knowledge of programming in Visual Basic is assumed.

### The IActiveViewEvents Interface

As described in the ArcObjects Developer Help, the IActiveViewEvents interface "provides access to events that occur when the state of the active view changes." The active view in this instance can refer to the Map (also known as Data Frame view in ArcMap) or the PageLayout (Layout view in ArcMap), as well as numerous other co-classes.

Tracking changes to the active view allows the developer to detect when certain processes have been run and take appropriate action. For example, the developer might have created a form that contains the list of layers in the map. By detecting when the user adds a new layer to the map, the developer can choose to update the list of layers on the form.

### Using the WithEvents Keyword to Track Events on the Layout

Variables may be declared with the optional WithEvents keyword. Declaring with this keyword indicates that these variables are able to respond to events triggered by other objects. The code in Listing 1 will declare a module-level variable on the PageLayout coclass. The WithEvents keyword will cause it to respond to events occurring on the layout. (A module-level variable is also known as a global variable and is available in other subroutines rather than being limited to a particular subroutine).

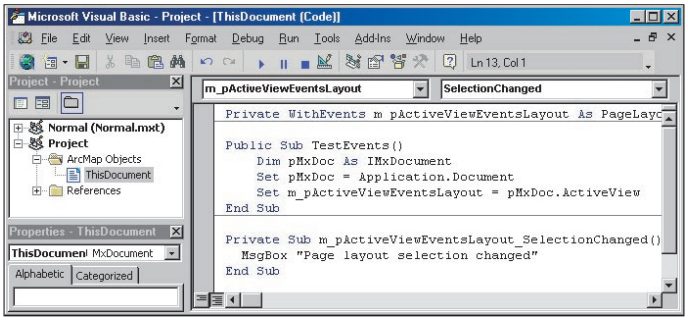
```
Private WithEvents m_pActiveViewEventsLayout As
PageLayout

Public Sub TestEvents()
  Dim pMxDoc As IMxDocument
  Set pMxDoc = Application.Document
  Set m_pActiveViewEventsLayout = pMxDoc.ActiveView
End Sub
```

**Listing 1**

```
Private Sub m_pActiveViewEventsLayout_
SelectionChanged()
  MsgBox "Page layout selection changed"
End Sub
```

**Listing 2**




*Open the VBA Editor using > Tools > Macros > Visual Basic Editor and paste the code snippets in Listings 1 and 2 in the VBA Editor under Project > ArcMap Objects > ThisDocument.*

The line "Set m_pActiveViewEventsLayout = pMxDoc.ActiveView" in Listing 1 shows that the global variable m_pActiveViewEventsLayout will track events on the map document's ActiveView, which is the layout window. These events can be captured using members on the IActiveViewEvents interface such as the SelectionChanged member shown in Listing 2.

To test this code

1. Start a new ArcMap session.
2. Open the VBA Editor using Tools > Macros > Visual Basic Editor.
3. Paste the code snippets in Listings 1 and 2 in the VBA Editor under > Project > ArcMap Objects > ThisDocument.
4. Add some data to the map.
5. Switch to Layout view by choosing View > Layout View from the Standard menu.
6. Run the TestEvents macro under Tools > Macros > Macro to initialize the variables.
7. Select or unselect the Data Frame on the layout.

This should trigger the display of a message box indicating that the Selection Changed event has occurred on the active view (i.e., the PageLayout object). In this case, the subroutine m_pActiveViewEventsLayout_SelectionChanged was run automatically when the layout's selection set changed. This code will run every time the selection set changes on the layout.

### Tracking Events on the Data Frame

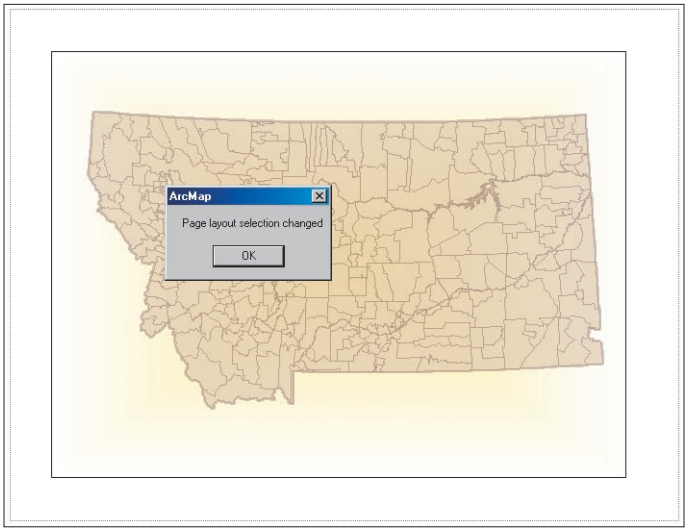
A similar method can be used to monitor events on the Map (i.e., Data Frame view in ArcMap). To test events on the Data Frame, modify the code shown in Listing 1 written earlier to add the lines shown in bold in Listing 3.

Variables may be declared with the optional WithEvents keyword. Declaring with this keyword indicates that these variables are able to respond to events triggered by other objects.

```
Private WithEvents m_pActiveViewEventsLayout As
PageLayout
Private WithEvents m_pActiveViewEventsMap As Map

Public Sub TestEvents()
    Dim pMxDoc As IMxDocument
    Set pMxDoc = Application.Document
    Set m_pActiveViewEventsLayout = pMxDoc.ActiveView
    Set m_pActiveViewEventsMap = pMxDoc.FocusMap
End Sub
```

**Listing 3**




*This code declares a module-level variable on the PageLayout coclass. Because the WithEvents keyword was used, it will respond to events occurring on the layout.*

In this case, rather than declaring the global variable on the PageLayout coclass, it is declared on the Map coclass. The variable is then set against the map document's FocusMap, which is the active Data Frame.

Now paste the code in Listing 4 into the VBA document, replacing code shown in Listing 2, to trap events occurring against the map. This sample code also tracks the SelectionChanged event, but because it uses the global variable defined against the Map, it works against the Data Frame rather than the Page Layout.

```
Private Sub m_pActiveViewEventsMap_
SelectionChanged()
  MsgBox "Data frame selection changed"
End Sub
```

**Listing 4**

Reinitialize the variables by running the setup script under Tools > Macros > TestEvents. Then select some map features using the Select tool. This should trigger the display of a message box indicating that the Data Frame's selection set has changed. Unselect the Data Frame on the layout using the Pointer tool, and note that the Page Layout selection event fires as before and displays the "Page layout selection changed" message box.

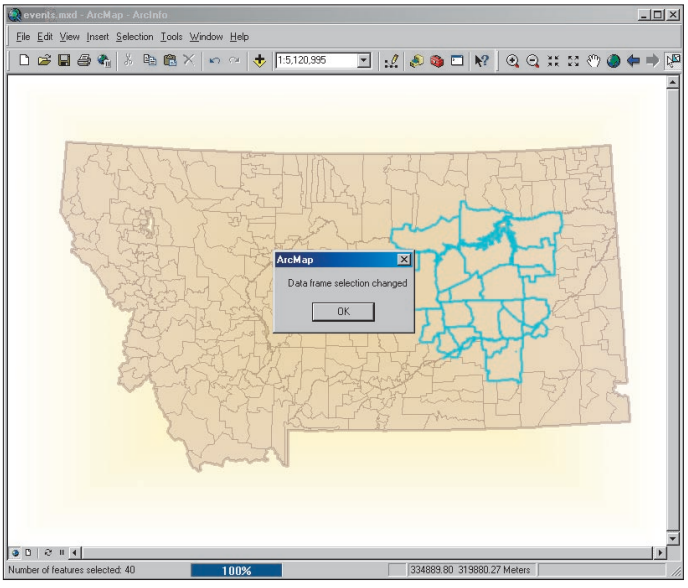
TestEvents illustrates that the IActiveViewEvents interface is used by both the PageLayout and the Map but is implemented slightly differently. In the Data Frame, event is triggered by changes to the feature selection, while on the Page Layout, the event is triggered by changes to the map elements selection.

### Changes to the FocusMap

1. In Layout view, add a new Data Frame by choosing Insert > Data Frame from the Standard menu. Add a dataset to this new Data Frame. If necessary, reposition the Data Frames such that they are both visible.
2. The Page Layout selection changed message box should appear when adding the new Data Frame because the new Data Frame is selected when it is added. If it does not appear, rerun the TestEvents script to reinitialize the variables.
3. Select a feature on the new Data Frame using the Select tool and note that the "Data frame selection changed" message box does not appear. Select a feature in the original Data Frame, and the message will appear as expected.

This behavior occurs because the original IActiveViewEvents variable referenced the application's FocusMap. FocusMap changes when a new Data Frame is added to the map. This provides a good example of the

*Continued on page 46*




*Because the script uses the global variable defined against the Map, it works against the Data Frame rather than the Page Layout and triggers the message when the Data Frame selected changes.*

# Event-Driven Programming Using IActiveViewEvents

*Continued from page 45*

usefulness of programming against IActiveViewEvents.

Copy and paste the code in Listing 5 into a new subroutine. It will execute when the FocusMap changes. When the FocusMap changes, the code in Listing 5 will reinitialize the global Map variable against the current FocusMap, which is the new active Data Frame. To test these changes, run the macro Tools > Macros > TestEvents and then experiment with changing the feature selection set in both Data Frames and selecting and unselecting the Data Frames in the Page Layout. Change in the selection status should be reported in both Data Frames as well as within the layout.

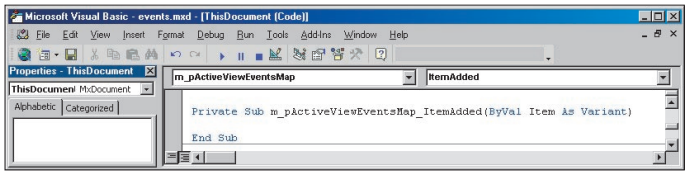
```
Private Sub m_pActiveViewEventsLayout_
FocusMapChanged()
  Dim pMxDoc As IMxDocument
  Set pMxDoc = Application.Document
  Set m_pActiveViewEventsMap = pMxDoc.FocusMap
  MsgBox "Page layout focus map changed"
End Sub
```

**Listing 5**

### Other IActiveViewEvents Members

So far, the sample code above has used the SelectionChanged and FocusMapChanged members. A list of events on IActiveViewEvents that can trigger an action is shown in the accompanying table. Each member can be captured in a manner similar to that used in the previous example.

To simplify this process, choose m_pActiveViewEventsMap on the left-hand panel of the VBA Editor and choose the drop down on the right-hand side. Choosing any members in the drop-down list will create a code stub for that event. For example, choosing ItemAdded from the drop-down list will create the declaration portions for the script that will run when a new item is added to the map. The developer can add code that will be run whenever the ItemAdded event occurs. As many of these members can be implemented as are required by a particular project.




*To simplify this process, choose m_pActiveViewEventsMap on the left-hand panel of the VBA Editor and choose the drop down on the right-hand side. Choosing any members in the drop-down list will create a code stub for that event. For example, choosing ItemAdded from the drop-down list will create the declaration portions for the script that will run when a new item is added to the map.*

### Practical Uses for IActiveViewEvents Programming

The following examples use scripts from the ArcObjects Developer Kit and the ArcScripts Web site to accomplish useful tasks using IActiveViewEvents programming.

#### *Keeping track of layers on the map*

As previously stated, the IActiveViewEvents interface can be useful in keeping a list of map layers up to date on a form. This is the approach taken in the Swipe Tool sample script provided in the ArcObjects Developer Kit. (The Swipe Tool is implemented in the default interface in ArcGIS 9.1.) In this sample, a list of raster layers is provided in a combo box on a form. When a layer is added to or deleted from the map, the list of layers in the combo box needs to be updated. This is achieved by monitoring the IActiveViewEvents interface on the Map and calling a subroutine to refresh the layer list when the contents change. The relevant portions of the script are shown in Listing 6.

| Member | Description |
|---|---|
| AfterDraw | Fired after the specified phase is drawn. |
| AfterItemDraw | Fired after an individual view item is drawn. Example: View items include layers in a map or elements in a Page Layout. |
| ContentsChanged | Fired when the contents of the view changes. |
| ContentsCleared | Fired when the contents of the view is cleared. |
| FocusMapChanged | Fired when a new map is made active. |
| ItemAdded | Fired when an item is added to the view. |
| ItemDeleted | Fired when an item is deleted from the view. |
| ItemReordered | Fired when a view item is reordered. |
| SelectionChanged | Call this function to fire the selection changed event. |
| SpatialReferenceChanged | Fired when the spatial reference is changed. |
| ViewRefreshed | Fired when view is refreshed before draw happens. |

*Events on IActiveViewEvents that can trigger an action*

```
Private WithEvents ActiveViewEvents As Map

Private Sub ActiveViewEvents_ContentsChanged()
    ' call a script to refresh the list on the form
    frmSwipeLayers.RefreshList
  Exit Sub
```

**Listing 6**

The list of layers is also updated when a layer is added or deleted or the order of the layers changes. This ensures that the list of layers on the form follows the same order as the layers in the Table of Contents. This script is available in the ArcObjects Developer Help under Samples > Map_Analysis > Tools > Swipe Tool.

#### *Drawing dynamic feature buffers*

Another script included in the ArcObjects Developer Help is the Draw Polygon Buffers sample. This code will create a dynamic buffer around the currently selected polygon feature. This is achieved by monitoring the IActiveViewEvents interface on the map and setting the global variable against the FocusMap. Any changes to the feature selection set within the Data Frame cause the ActiveViewEvents_SelectionChanged member to fire. Code within this subroutine creates a buffer around the selected feature. The relevant sections of the developer's sample code are shown in Listing 7. Locate this script by accessing the ArcObjects Developer Help and choosing Samples > ArcMap > Draw Polygon Buffers.

```
' declare the global variables
Private WithEvents ActiveViewEvents As Map
Private m_pMxDoc As IMxDocument

' initialise the IActiveViewEvents variable
against the data frame
Public Sub InitEvents()
  Set m_pMxDoc = Application.Document
  Set ActiveViewEvents = m_pMxDoc.FocusMap
End Sub

' create the buffer around the selected feature
Private Sub ActiveViewEvents_SelectionChanged()

End Sub
```

**Listing 7**

### *Events Listener Sample Application*

A sample application called Events Listener, available on the ArcScripts page, can be used to "listen" for events on the MXD and the IActiveViewEvents interface. This application displays a form that lists the events as they occur and can be useful in deciding where to place custom code to cause it to run after the desired event has occurred. Locate this script by visiting www.esri.com/arcscripts and searching on the keywords "event listener."

### Further information

See the ArcObjects Developer Help under IActiveViewEvents Interface for more detailed information including a full description of the interface members and the other coclasses that support this interface. This document has focused on just the Map and PageLayout coclasses but there are many more events that can be trapped using this interface. Send any problems, questions, or comments to Stephen Lead, Technical Solutions Group, ESRI (UK) Ltd., at slead@esriuk.com.