Migrating Widgets to the ArcGIS Viewer for Flex

Some tips and tricks to ease the process

By Robert Scheitlin, Calhoun County; and Bjorn Svensson and Derek Law, Esri

This article tells developers how to migrate widgets developed with the Sample Flex Viewer (SFV) to the ArcGIS Viewer for Flex. It assumes readers have experience using the ArcGIS API for Flex and are familiar with the Adobe Flash development environment, and experience developing with the Flex API and Adobe Flash is strongly recommended.

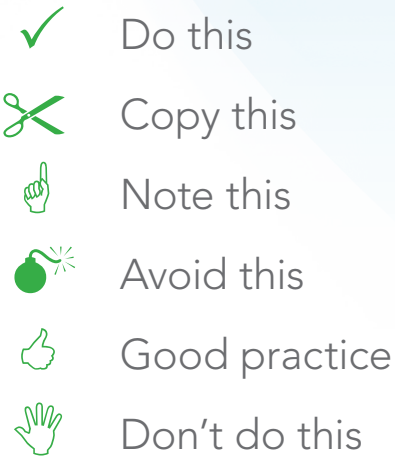
Released in November 2008, the SFV was a developer sample built on the ArcGIS API for Flex. It enabled nonprogrammers to deploy a rich Internet application for ArcGIS Server with minimal effort. Since its release, the SFV has been downloaded more than 30,500 times, and many sites have been built on the SFV.

SFV also provided a framework for Flex API developers to customize and extend the viewer. One important area was the ability to create custom widgets. Widgets are modular pieces of code that extend or add to the functionality of the SFV. They can be tailored by the widget developer for specific tasks that require particular data and conditions to be present in the viewer, or they can be generic and allow SFV to be configured by nonprogrammers to work with their own data. More than 50 widgets have been created for SFV and shared on the ArcGIS API for Flex code gallery. Many of these widgets can still be accessed from the Esri ArcScripts site (arcscripts.esri.com/; search for "flex" AND "viewer").

In September 2010, Esri released the ArcGIS Viewer for Flex—the official product release of the SFV. It includes 20 core widgets that support many standard web mapping application functionalities.

Many users wondered about the SFV widgets that were produced and shared on the ArcGIS API for Flex code gallery. Would these widgets "just work" in the new ArcGIS Viewer for Flex application?

Unfortunately, the answer is no. The ArcGIS Viewer for Flex uses a framework that differs from SFV. It is based on a newer release of the ArcGIS API for Flex and utilizes the latest Adobe Flash technology. To use widgets previously developed for the SFV, the code base for those widgets must be migrated and recompiled for the new ArcGIS Viewer for Flex 2.x API libraries.



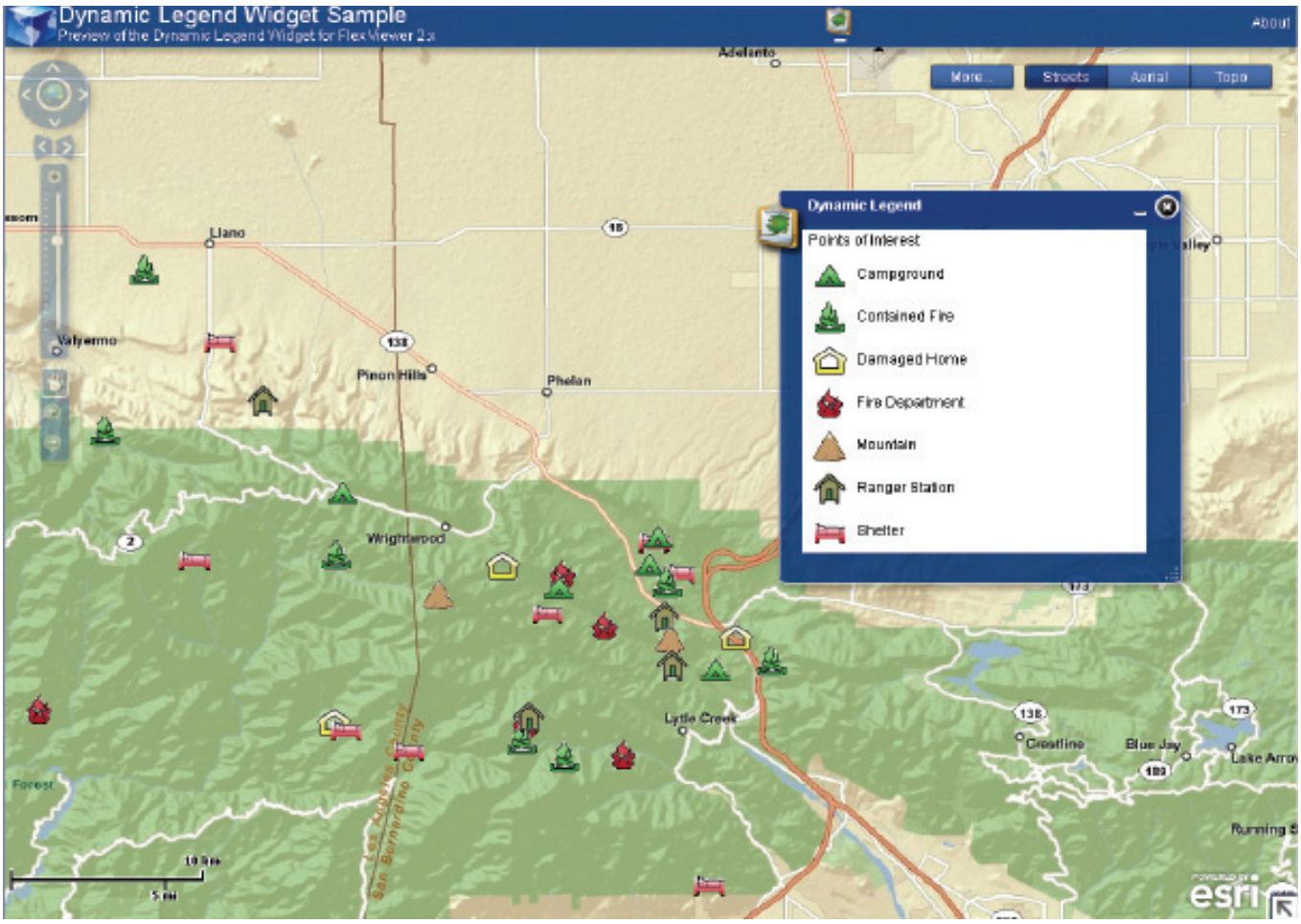
This article provides tips and recommended practices to help Flex developers easily migrate custom widgets from the SFV to the ArcGIS Viewer for Flex. Flex developers should be aware of the differences between the SFV and the ArcGIS Viewer for Flex. These differences are summarized in Table 1.

Sample Flex Viewer	ArcGIS Viewer for Flex
Uses Adobe Flex SDK v3	Uses Adobe Flex SDK v4
ViewStack	States
mx:Text	s:Label
mx:HBox	s:VGroup
mx:ComboBox	s:DropdownList
Based on ArcGIS API for Flex 1.x	Based on ArcGIS API for Flex 2.x
symbol package	symbols package
com.esri.ags.tasks.Query	com.esri.ags.tasks.supportClasses.Query
com.esri.ags.tasks.FeatureSet	com.esri.ags.FeatureSet
Framework	Framework
WidgetEffects	viewer:transitions
com.esri.solutions.flexviewer.SiteContainer	com.esri.viewer.ViewerContainer
BaseWidget	viewer:BaseWidget

↑ Table 1: ArcGIS Viewer for Flex equivalents for SFV

Table 1 highlights some of the subtle—but key—changes in development patterns between the SFV and the ArcGIS Viewer for Flex. The concepts are the same, even though they may have been renamed. However, this is not a comprehensive list of these differences. At the Adobe software development kit (SDK) level, Adobe recommends using the new Spark components. For example, s:VGroup is used instead of mx:HBox. For more detailed information, see the resources list at the end of this article.

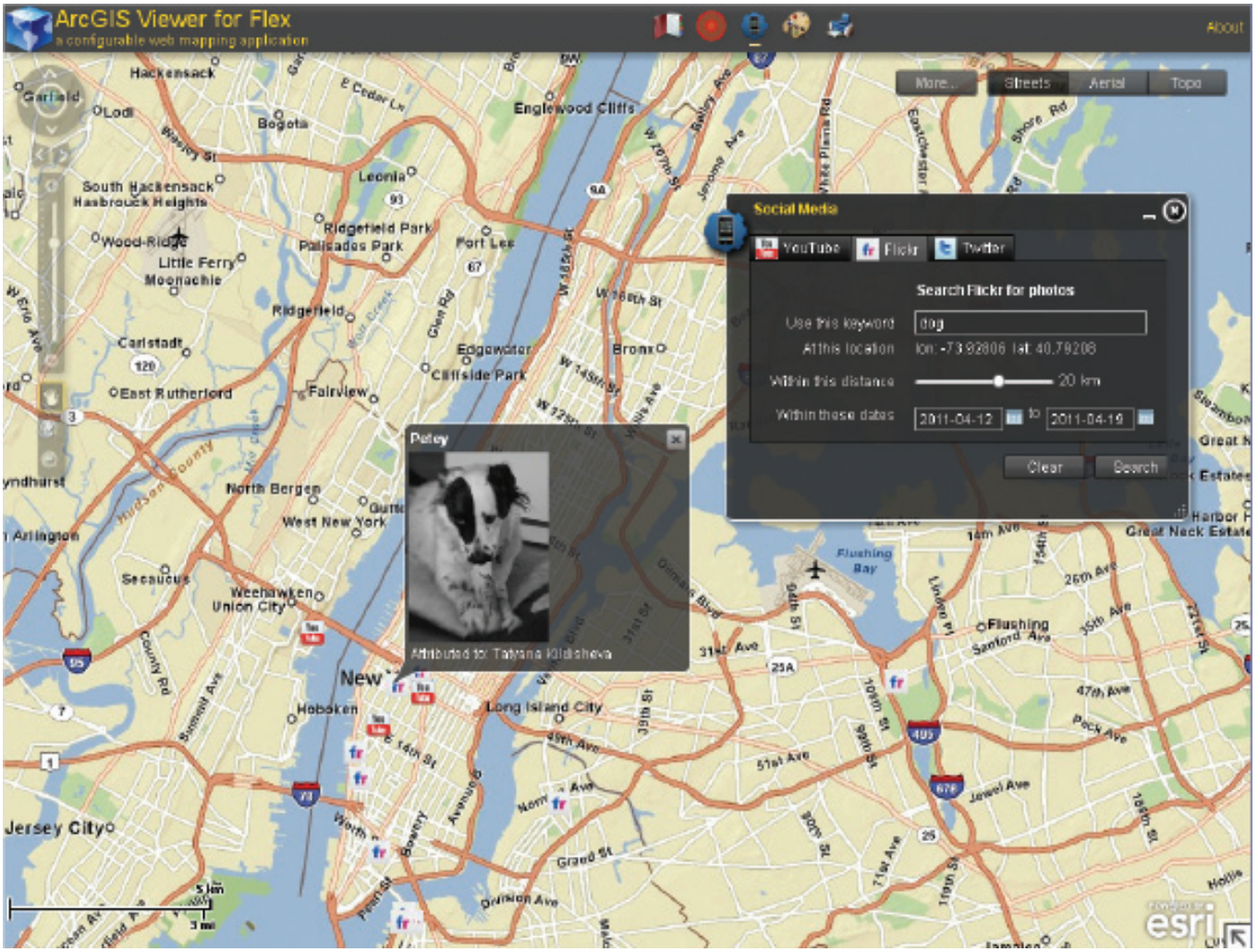
↓ This Dynamic Legend widget created by Robert Scheitlin modifies the contents of the legend based on scale dependency and layer visibility. Map services and/or specific layers in a map service may also be excluded from the legend.



General Tips on Widget Migration

- ✓ Start the widget migration process with a new MXML Component. Create the new MXML Component as part of a package in the widgets folder (i.e., widgets.LiveLayer). Follow Viewer coding standards. The widget package should share the same name as the widget name (minus the word "widget"). For example, the full widget name and package would be widgets.LiveLayer.LiveLayerWidget. Base the new MXML Component on BaseWidget.
- ✋ Don't give the new component a width or height; that is handled in the widget template.
- ✓ If the widget is going to reference custom components, such as item renderers and data groups (which are designed to replace the mx:Repeater in the new Adobe Flex SDK 4 environment), add the widgets.xml name space to the BaseWidget element (e.g., xmlns:livelayer="widgets.LiveLayer.*).
- ✓ Use the widgetConfigLoaded event if the widget has a configuration file. This ensures that the widget configuration file has been loaded before you try to use it. Having a widget configuration file allows nondevelopers to change certain aspects of the widget without altering the code and compiling the application.
- ✓ An fx:Script block is needed for the ActionScript code that will be migrated from an mx:Script block in the old widget. Add an fx:Script block by typing it in the new widget file instead of just copying the mx:Script block from the old widget.
- ☝ States replace the ViewStacks used to separate pages or views in old widgets. A little planning will go a long way here. Examine the old widget and determine how many VBox elements are children of the ViewStack, that is, how many states will be needed. Each state must have a name as shown in the example in Listing 1.

↓ The Social Media widget created by Ping Jiang searches YouTube for videos, Flickr for photos, and Twitter for Tweets based on a keyword.



```
<viewer:states>
  <s:State name="filterResults"/>
  <s:State name="resultsList"/>
</viewer:states>
```

↑ Listing 1: Each state must have a name.

✓ In the old SFV, an animation that occurred when moving from one view to another was handled by a custom ActionScript class called WidgetEffects. In the new viewer, transitions are used for this purpose. The targets for transitions will be the names of the states defined previously. An example is shown in Listing 2.

```
<viewer:transitions>
  <s:Transition autoReverse="true" toState="*">
        <s:Fade targets="{[filterResults, resultsList]}"/>
  </s:Transition>
</viewer:transitions>
```

↑ Listing 2: Handle animations between views with transitions.

✂ Copy the MXML elements that define the UI of the old widget, paste them into the new widget, and comment them out. The commented old code can serve as a reference. This will save some time because it eliminates the need to switch back and forth from old widget code to new—both versions will be present.

Moving the widgets' MXML code from mx components to Spark components during the code migration is recommended. Use Table 1 to determine the Spark equivalents for some mx components. The WidgetTemplate element is still the base for the widget's UI. The new widget template in the ArcGIS Viewer for Flex has renamed a few of the events. For example, the "widgetClosed" event is now just "closed" and the "widgetOpened" event is now "open."

The height and width of a widget's UI is defined in the widget template. Each widget state will be a Spark group element, and each ID will share the same name as the states defined earlier. Set the visibility of the group to false and add another attribute, "visible." After the attribute "visible" is typed, add a dot after it, and the automatic code completion will display the available states (when using the Adobe Flash Builder IDE). Choose the state name of the current group.

```
<viewer:WidgetTemplate id="wTemplate"
                width="430" height="240"
                open="widgetOpenedHandler(event)"
                closed="widgetClosedHandler(event)"
                minimized="widgetMinimizedHandler(event)">
  <s:Group id="resultsList"
         width="100%" height="100%"
         visible="false"
         visible.resultsList="true">
       <s:layout>
              <s:VerticalLayout gap="10" horizontalAlign="center"
verticalAlign="middle"/>
       </s:layout>
  </s:Group>
  <s:Group id="filterResults" width="100%" height="100%"
         visible="false" visible.filterResults="true">
       <s:layout>
              <s:VerticalLayout gap="4" horizontalAlign="center"
verticalAlign="middle"/>
       </s:layout>
  </s:Group>
</viewer:WidgetTemplate>
```

↑ Listing 3: Widget template

- Examine the code for old mx components that could be "Sparkified." For example, if the old code uses an mx:Text, then its Spark counterpart is s:Label; an mx:HBox could become an s:Hgroup, mx:Button could become s:Button, and mx:ComboBox could become an s:DropDownList.
- Another practical tip is to copy all the old mx:Script code from the old widget and paste it inside the new fx:Script block. As mentioned earlier, don't copy the mx:script block in its entirety; just copy the contents between the <![CDATA[ ]]>. There will be several errors that will have to be addressed one at a time.
- Replacing the import statements in the script block that have changed in the ArcGIS API for Flex 2.2 is important.

SFV import statement	Replacement import statement in ArcGIS API for Flex 2.2
`import com.esri.solutions.flexviewer.SiteContainer`	`import com.esri.viewer.ViewerContainer;`
`import com.esri.ags.symbol.*`	`import com.esri.ags.symbols.*;`
`import com.esri.ags.tasks.Query`	`import com.esri.ags.tasks.supportClasses.Query;`
`import com.esri.ags.tasks.FeatureSet`	`import com.esri.ags. FeatureSet.`

↑ Listing 4: Replace import statements.

One simple way to fix these is to examine the reported compile error. Double-click it to go to the specific line; put the cursor at the end of the offending class; and press Ctrl+Spacebar for Content Assist, which will add the required import statement.

- While it is not required, it *is* a good practice to migrate mx:Repeater to the Spark DataGroup class. Accomplishing this involves creating three new items, *Results.as, *ResultDataGroup.as, and *ResultItemRenderer.mxml. Fortunately, there are several examples of this code in the ArcGIS Viewer for Flex. A quick shortcut: simply copy and paste these three items from SearchWidget and rename them with the new widget's name.
- If the old widget used an mx:Repeater, the code probably has many references to its dataProvider. It will be necessary to create a bindable private var of type ArrayCollection to replace it. Everywhere in the code that references the repeaters, dataProvider must be changed to reference this new ArrayCollection.
- The new ArcGIS Viewer for Flex allows developers to specify a custom info window to use with a particular widget or one of the widget templates that comes standard with the viewer. Using this new capability involves several code additions and changes, such as overriding the widget's showInfoWindow function. Rather than identifying each line that must be changed and added in this article, look at one of the existing core Viewer widgets and search for "info." That search will return items like the infoURL string, which holds the infoURL string from the widget configuration file, or the DATA_CREATE_INFOWIDGET event.
- When using the new info window and data group (when replacing mx:Repeater), a couple of new import statements must be added:

```
import com.esri.viewer.IInfowindowTemplate;
import mx.core.UIComponent;
import spark.components.supportClasses.ItemRenderer; and
import com.esri.viewer.AppEvent.
```

- If the data group and item renderer will be updated, the mouseOverRecord, mouseOutRecord, and clickRecord event handlers must also be updated to convert events passed to these handlers to an itemRenderer instead of using the infoData object.

About the Authors

Robert Scheitlin is the GIS manager for Calhoun County, Alabama. A GIS software developer for 12 years, he has worked on projects that included full ArcObjects custom applications, ArcGIS Engine applications, and ArcGIS Server API for Flex and Flex Viewer applications. He has used and customized the Sample Flex Viewer since its release and supported Flex developers on the ArcGIS API for Flex forum. After initially focusing on Visual Basic and Visual Basic. NET, he is now focusing primarily on Flex. His background as an Esri Authorized Instructor has given him the ability to teach others about software development and customization.

Bjorn Svensson is the lead product engineer for ArcGIS API for Flex and ArcGIS Viewer for Flex. He has worked with web mapping at Esri for 10 years. Previously he worked as a GIS consultant in Africa, Asia, Europe, and the Americas.

Derek Law is part of the ArcGIS Server product management team, covering the ArcGIS Viewer for Flex and Silverlight. He has been with Esri for 10 years, with extensive experience working with geodatabases and ArcSDE technology. In recent years, his focus has been on the configurable client viewers for ArcGIS Server.

```
var llResult:LiveLayerResult = ItemRenderer(event.target).data as
LiveLayerResult;
```

- When migrating widget code and using the queryTask, if the code is not connecting to an instance of ArcGIS Server 10 or higher, you need to set queryTask.useAMF = false.
- Title bar buttons in the new ArcGIS Viewer for Flex no longer return events, so the click handler does not require an event.

Old format

```
private function toggleFilterPanel(event:MouseEvent):void
```

New format

```
private function showResultsList():void
```

- The order in which title bar buttons are added is the opposite order in which they were added in the SFV (e.g., the first button to appear on the left should now be the last one added).
- The third property for the addTitlebarButton function is used to designate whether the button is selectable. The default value is true.
- The assets directory in the SFV was com/esri/solutions/flexviewer/assets/images/icons/. The assets directory in ArcGIS Viewer for Flex is located at assets/images/. (Notice there is no subfolder of icons.)

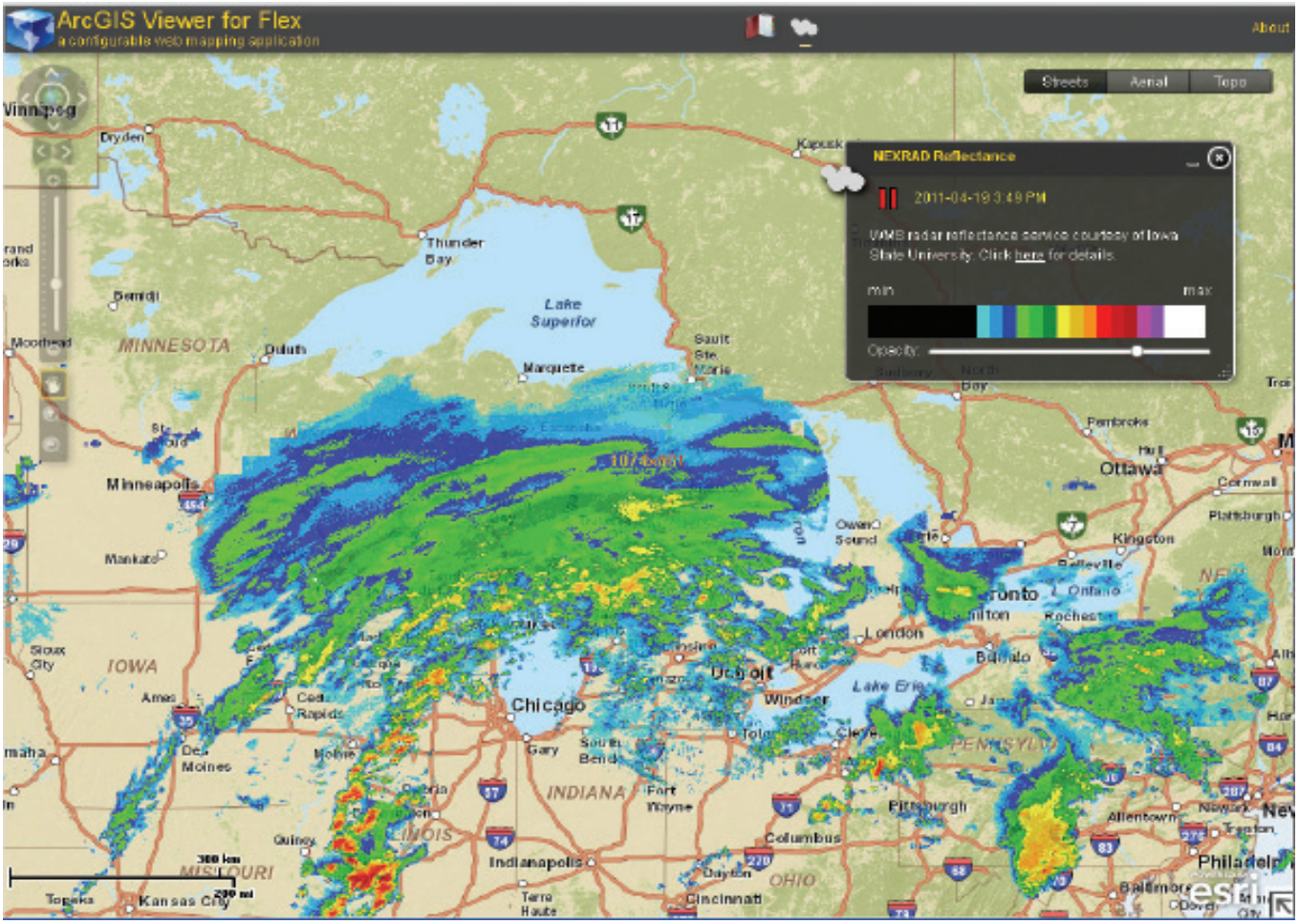
To summarize, there are many key items that Flex developers should be aware of when migrating a custom widget from the Sample Flex Viewer to the ArcGIS Viewer for Flex. An example of migrated widget code can be found at gis.calhouncounty.org/DevSummit2011. It demonstrates the LiveLayerWidget code and includes developer comments.

Online Resources

Differences between Adobe Flex SDK 3 and Adobe Flex SDK 4
adobe.com/devnet/flex/articles/flex3and4_differences.html

Esri API changes between ArcGIS API for Flex 1.x and 2.x
http://help.arcgis.com/en/webapi/flex/help/index.html#/Migrating_from_1_3_to_2_0/017p0000000z000000/

ArcGIS Viewer for Flex Resource Center
links.esri.com/flexviewer



↓ Mark Deaton's widget shows a changing series of NEXRAD radar reflectance images (indicating severe weather) over the US for the previous hour. It also demonstrates the use of WMS layers via the ArcGIS Flex API.