Closest Facility Widget for Esri ArcGIS Viewer for Flex

# Introduction

Here are instructions for using the accompanying widget source and object code files for performing closest-facility analyses in Esri ArcGIS Viewer for Flex (API versions 2.1 – 3.6). 3.0 - 3.6 also include a meta.xml file to allow them to be used in the Application Builder. See the end of this document for more information.

# Configuration options

First, make sure you have available a running instance of a closest facility network analysis service. You may use one running on Esri’s sample server, which is preconfigured in the supplied widget’s configuration file; or you may set up and use your own. If you use your own, you will need to deploy a cross-domain permissions file (crossdomain.xml) to the root directory of the web server hosting the elevation service. There’s more information on crossdomain.xml here:

<http://help.arcgis.com/en/webapi/flex/help/content/deploy_application.htm>

You’ll also need a facilities dataset supplying facilities as points in a feature layer. This distribution is configured to use data hosted on Esri’s sample server, but you may substitute your own. You may also configure the widget to read in event features from a different feature layer. This is optional and not configured by default, since the widget lets you click to place events on the map.

# Installation into Viewer instance

You may install and use the widget without loading it into the Flash Builder development environment or compiling it. This section describes how to add the supplied, compiled modules into an existing instance of the Esri ArcGIS Viewer for Flex 2. These instructions assume you have a running instance of the application. You may obtain it here:

<http://help.arcgis.com/en/webapps/flexviewer/index.html>

Copy the supplied “widgets” subdirectory into your instance’s root directory. Note that there are two versions included: use the one underneath “**runtime (2.1 - 2.3.1)**” if you’re using FlexViewer 2.1 through 2.3.1; otherwise, use the one under “**runtime (2.4)**” if you’re using FlexViewer 2.4 or “**runtime (2.5)**” if you’re using FlexViewer 2.5. Do not use the one underneath “development” unless you’re working with the project in the Flash Builder development environment (see the next section).

Now you’ll need to make sure there’s an entry for the widget in the Viewer’s main config file. The easiest way is to use the one supplied underneath “runtime”: copy it over your Viewer’s “config.xml” file.

Note that the only way the supplied config.xml differs from the one distributed with the Viewer is the addition of the following entry in the “widgetcontainer” section:

For versions 2.1 through 2.4:

<widget label="Closest Facilities" icon="assets/images/i\_servicearea.png"

config="widgets/APL/ClosestFacility/ClosestFacilityWidget.xml"

url="widgets/APL/ClosestFacility/ClosestFacilityWidget.swf"

/>

Or for version 2.5:

<widget label="Closest Facilities" icon="assets/images/i\_servicearea.png"

config="widgets/ClosestFacility/ClosestFacilityWidget.xml"

url="widgets/ClosestFacility/ClosestFacilityWidget.swf"

/>

Now that config.xml points to the new widget, open the Viewer in a browser. You may need to refresh the browser cache and reload to see the widget appear.

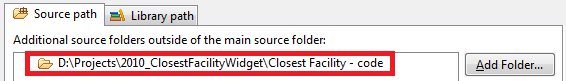
Click the green dot icon at the top, then click on the map to place an event you want to route to nearby facilities. Click the blue polyline icon at the top, then draw a line on the map to place a barrier routes cannot cross (each click places a new vertex; double-click to finish the barrier). Then click “Solve” to submit the problem to the server. The route solution should draw on the map after a few moments.

# Installation – Source Code

Source code for this widget is available under the supplied directory “development”. This code is not a standalone project, but is meant for inclusion into an existing ArcGIS Viewer for Flex 2 project. First, make sure your Viewer project is open in Flash Builder. Extract the contents of the zip file to a location on your disk you will use in the next step.

You’ll need to add the source-code directory to the project build path. Open the project properties in Flash Builder; then choose the “Flex Build Path” item. Under the “source path” tab, click “Add Folder…” and select the “development\code” subdirectory you extracted from the zip file.

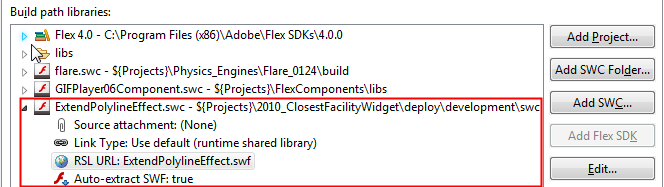
Once you click OK, it should show up in your project’s build path list:



Next, you must add the widget to the project’s modules list using the Project Properties > Flex Modules dialog page. You’ll need use the “Add…” button and dialog and browse for the “development\code\widgets\APL\ClosestFacility\ClosestFacilityWidget.mxml” file you extracted from the zip file.

You can choose to have this modules optimized for the index.html project, or compile it unoptimized. Since it probably won’t be useful in any other application, it’s better to optimize it.

You must also add the swc for the line-animation effect. In project properties, choose “Flex Build Path” and the “Library path” tab. Click “Add SWC…” and browse for the “development\swc\ExtendPolylineEffect.swc” (or “GeometryEffects.swc”) file you extracted from the zip file.



You can choose for the effect to be compiled into the main Viewer code, or left as a separate .SWF at runtime. No matter which approach you should take, it should behave the same at runtime.

# Configuration File

The widget’s configuration file is located in the same directory as the .swf or .mxml file. It’s quite basic, containing only a very few options.

* facilities|url: The URL for the ArcGIS Server feature layer serving the facilities points for the widget
* incidents|url: The (optional) URL for the ArcGIS Server feature layer serving incident or event points for the widget
* closestFacilitySvc|url: The URL for the ArcGIS Server network analyst closest-facility service providing routing for the widget
* symbology|routeRendererAttrName: The attribute field in the returned route results that will be used by the renderer to color the routes and set their widths
* symbology|routes: Instructions to the widget on how to render the various routes returned from the analysis
* symbology|barrier: Instructions to the widget on how to display the user-drawn barrier line
* symbology|event: Instructions to the widget on how to display the event features drawn by the user and loaded from the incidents feature service
* symbology|animateRoutesDuration: Instructions to the animation effect on how many milliseconds to take to draw the returned route lines
* routeRequestAccumulatedAttributes: A list of <attrName> records specifying which attributes in the streets network dataset you want accumulated in the returned routes. Remove all <attrName> entries if you don’t want any accumulation performed. Commonly you’ll want to accumulate length or time. Valid accumulation attributes will be summed up for each result route and placed in an attribute called “Total\_[name]”: for example, an accumulation attribute of “Length”, if valid, will be summed as “Total\_Length” in the result routes.

# Addendum: Usage with Viewer version 3

The distribution file now includes a sub-archive, “3.0 (AppBuilder compatible)”. This will need to be extracted from its containing archive and saved to your disk as a standalone .zip file. Then the Viewer Application Builder’s “Manage Widgets” utility can be pointed to this .zip file in order to import the widget. That archive contains the source code and compiled widget as well; you may use them with Flash Builder and Viewer 3 just as with version 2.x, as described above.