ArcLogistics Tutorials: How to create a Widget plugin

This document explains how to create a *Widget* type of plugin for ArcLogistics. Before starting plugin development, please read the document titled **ArcLogistics Plugins: An Introduction**, to have a firm understanding of the basic concepts, and the document titled **ArcLogistics Plugins: Getting Started**, for a step-by-step guide for creating a simple "Hello World" plugin.

Setting up the project environment

1. Start Visual Studio, and create a new C# project of *WPF User Control Library* type. Change the target framework to **.Net Framework 4.0** if it is not already selected.

2. To make debugging easier, change the *Build* settings by providing the **ArcLogistics install path** as the *Output path* for the plugin. This will cause the plugin dll to be built in the install directory, where it is automatically activated and is ready to be used.

3. Also, change the *Start Action* setting under the *Debug* tab to **Start External Program** and select the **ESRI.ArcLogistics.App.exe** file from the install directory. This will launch ArcLogistics with the plugin enabled when you start debugging in Visual Studio.

Implementation

1. Add references to the following two ArcLogistics files from the install directory: **ESRI.ArcLogistics.App.exe** and **ESRI.ArcLogisticsNG.dll**. You may need to add more references as your development progresses.

2. Use the following block of code as a guide to modify the xaml file in your project. The idea is to create a small page with a label in it. You can add more content inside the grid later to make the widget more useful.

```
<widgets:PageWidget x:Class="WidgetPluginTutorial.MyCustomWidge"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:widgets="clr-
       namespace:ESRI.ArcLogistics.App.Widgets;assembly=ESRI.ArcLogistics.App"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008" >
       <Grid>
              <Label Content="Name:" Height="28" HorizontalAlignment="Left"
                     Margin="10,10,0,0" Name="label1" VerticalAlignment="Top" />
       </Grid>
</widgets:PageWidget>
```

3. Add the following lines of code to the top of the implementation file (.cs):

```
using ESRI.ArcLogistics.App.Widgets;
using ESRI.ArcLogistics.App;
using System.Xml;
```

   You may need to add additional usings as your development progresses.

4. Add the following line of code just above the partial class declaration for the plugin:

```
[WidgetPlugIn(new string[1] { @"Schedule\OptimizeAndEdit" })]
```

This registers the new Widget with ArcLogistics, so that it appears on the *Optimize and Edit* action panel on the *Schedule* tab. Look at the table at the end of this document for a list of allowed values, and the locations they represent in the software.

5. Add the `PageWidget` class as a superclass for the widget, as shown in the following example:

```
public partial class MyCustomWidge : PageWidget
```

6. Right-click `PageWidget` and choose **Implement Abstract Class**. This adds stubs for one method and one property.

7. Modify the stubs in a manner similar to the code sample given below. The constructor initializes the controls of the widget. The *Initialize* method contains the code for initializing the contents of the widget. In this sample, we get the current temperature for Riverside, California from an RSS feed and show it. The *Title* Property specifies the text that appears on the widget.

```
[WidgetPlugIn(new string[1] { @"Schedule\OptimizeAndEdit" })]
public partial class MyCustomWidge : PageWidget
{
    public MyCustomWidge()
    {
        InitializeComponent();
    }
    public override void Initialize(ESRI.ArcLogistics.App.Pages.Page page)
    {
        label1.Content = "Data unavailable.";
        try
        {
            string URLString = @"http://www.weather.gov/xml/current_obs/KRAL.xml";
            XmlTextReader reader = new XmlTextReader(URLString);
            while (reader.Read())
            {
                if (reader.NodeType == XmlNodeType.Element && reader.Name ==
                                                                "temperature_string")
                {
                    if (reader.Read())
                        label1.Content = "Riverside, CA: " + reader.Value;
                    reader.Close();
                    break;
                }
            }
        }
        finally
        {

        }
    }

    public override string Title
    {
        get { return "Temperature"; }
    }
}
```

8. You can modify the initialize method, or add more methods to make the plugin more useful.

Testing

1. Build the solution. If build succeeds, the plugin dll should appear in the application install directory.

2. Either run ArcLogistics, or start debugging the solution from Visual Studio. In either case, the application will start, load the plugin and activate it automatically.

3. Once the application finishes starting up, the new widget will appear on the location you specified.

Wrapping up

Congratulations! You have just learned how to create *Widget* plugins for ArcLogistics. To learn how to create other kinds of plugins and adding more functionality to ArcLogistics, go through the other tutorials and various sample projects available on the *ArcLogistics Resource Center*. Refer back to the document titled **ArcLogistics Plugins: An Introduction** for a complete list and descriptions of all resources available for you to create your own plugins.

Appendix A: Allowed values for specifying Widget locations

Location	String to use
Getting Started Page	"Home\GettingStarted"
Projects Page	"Home\Projects"
License Page	"Home\License"
Fuel Types Page	"Setup\FuelTypes"
Locations Page	"Setup\Locations"
Vehicles Page	"Setup\Vehicles"
Drivers Page	"Setup\Drivers"
Mobile Devices Page	"Setup\MobileDevices"
Specialties Page	"Setup\Specialties"
Zones Page	"Setup\Zones"
Barriers Page	"Setup\Barriers"
Default Routes Page	"Setup\DefaultRoutes"
Schedule Page	"Schedule\OptimizeAndEdit"
Reports Page	"Deployment\Reports"
Export Page	"Deployment\Export"
General Preferences Page	"Preferences\General"
Map Display Preferences Page	"Preferences\MapDisplay"
Reports Preferences Page	"Preferences\Reports"
Routing Preferences Page	"Preferences\Routing"