ArcLogistics Plugins: An Introduction

This introduction is a great starting place for learning how to extend ArcLogistics by adding new custom functionality in the form of plugins. It explains below what a plugin is, and then describes the different types of plugins. Finally, it lists all the resources available for creating plugins.

What is an ArcLogistics plugin?

An ArcLogistics plugin is a software component that allows you to customize ArcLogistics and add special features or workflows that are not supported out of the box. Plugins can utilize the *ArcLogistics Plugin SDK*, which is based on .NET Framework 4.0. All classes and interfaces exposed by the API are located in the following two assemblies present in the application install directory:

- **ESRI.ArcLogisticsNG.dll**. This assembly contains the types which expose core ArcLogistics functions such as project management, data manipulation, routing, geocoding, export, licensing, etc. All components are UI-less so they can easily be used for creating out-of-process programs that integrate with ArcLogistics without using the UI of the main application.
- **ESRI.ArcLogistics.App.exe**. This executable assembly contains the types that allow a plugin to integrate directly into the application workflow in the form of custom extensions, pages, commands and widgets.

A plugin project must add references to these two ArcLogistics assemblies to consume their functionality. The output of the project must be a DLL, which can contain any number of individual plugins. When ArcLogistics starts up, it checks for the presence of plugin DLLs in two locations: the application install directory, and a user-specified location which can be set in the application's Preferences. If any compatible plugins are found, they are loaded and activated.

What are the different types of plugins?

There are four main types of Plugins:

- **Extensions**. These are plugins that are used for adding UI-less components to ArcLogistics. They are useful in situations where the requirement is to listen for some events to occur, and react to them by performing certain actions, without any user interaction.
- **Commands**. A Command represents an action to be performed, and can be presented to the user in the form of a button on a page's *Task* widget, or on a view's button group.
- **Pages**. The ArcLogistics application UI is presented to the user in the form of pages such as *Locations, Vehicles, Optimize and Edit, Export*, etc. A Page type of plugin represents a custom action panel with custom content and controls on it, and gets inserted into the application's tree of pages.
- **Widgets**. A widget represents a small piece of UI that is shown in the navigation pane on the left side of the application. Examples of widgets are *Quick Help, Tasks* and *Calendar*.

Figure 1 shows how these types of plugins fit into the ArcLogistics application architecture.

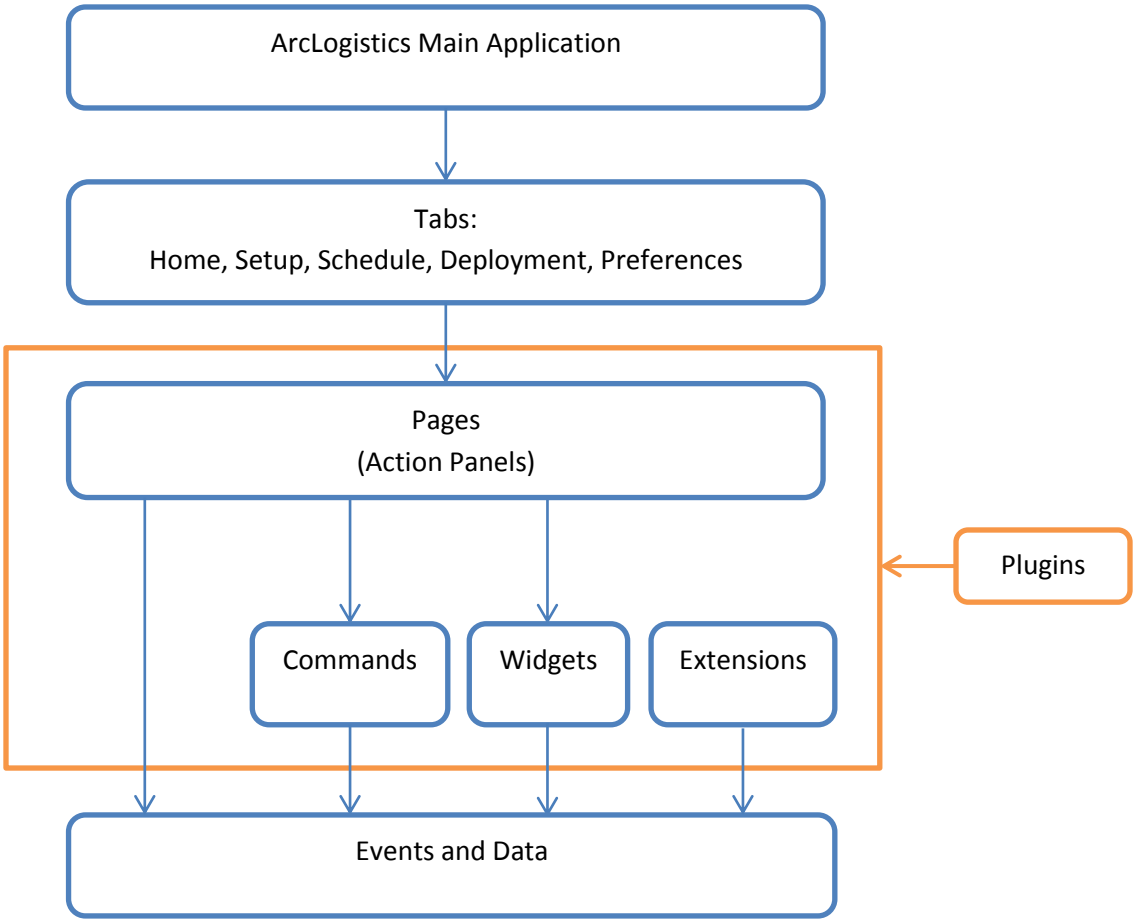

Figure 1: Conceptual architecture diagram.

How do I Start?

You can start creating your own plugins by first reading the document titled **ArcLogistics Plugins - Getting Started**. It describes the process of creating a simple "Hello World" plugin in a step-by-step manner.

There are various tutorials and sample projects available for download from the ArcLogistics Resource Center. Each sample contains the plugin dll, its source code and a document describing the various components of the source code. Find a sample which is similar to the plugin you are going to create, and use it as a starting point for learning how to finish your own plugin. These samples are written in C#, and are available as Visual Studio 2010 projects. The documents assume familiarity with C# and Visual Studio, so you may need a quick refresher if you haven't used them in a while.

The ArcLogistics Forum is the best place for posting queries, looking at answers to Frequently Asked Questions (FAQs), and interacting with other users, developers and Esri staff members.

Finally, the *ArcLogistics Plugin SDK Manual* contains detailed information about the ArcLogistics API, and is a great source for figuring out implementation details for plugins.