GRID PROTECTION ALLIANCE

Building New Analytics with Project Alpha

TSL Project Alpha

a complete time-series solution template

GPA User's Forum 2015

Atlanta, Georgia

What is Project Alpha?

- Project Alpha is a code solution “template” that provides a jump start to developing new products based on the Grid Solutions Framework’s Time-Series Library (TSL) technology.
- For example, this means that an Action adapter developed to run in the openPDC could be packaged independently into its own software package – with its own manager and setup applications.

What are the benefits?

- Fully customizable UI management system – colors, names, and menus.
- Ability to focus on the “analytic” and not the supporting framework.
- Installation of openPDC is *not* required.
- Application will be provisioned with all synchrophasor protocols, utility data formats (e.g., COMTRADE).
- Built-in support for reading and writing to OSI-PI and the openHistorian.

Where do I get it?

- The “Project Alpha” template contains all the source code, installation packages and tools needed to deploy a fully customized GSF Time-series Library application.
- The code can be downloaded from the following web site:

http://projectalpha.codeplex.com/

Will this project be limited to Windows?

- No – applications built using Project Alpha can be also deployed onto POSIX environments using Mono – the cross platform implementation of .NET. This means the analytics engine will also run on Linux and Mac OS X.
- Note that some tools, like the WPF based UI components (for example, the Project Alpha Manager used for configuration), will only run on Windows since Mono doesn't fully support this .NET API yet.

Will this require coding?

- Yes, some assembly required.
- We recommend you install a version of Visual Studio, at least version 2012 – there are free versions that should work fine, e.g., Visual Studio Express:

http://www.visualstudio.com/

Note that most "coding" will generally be restricted to the analytic code, i.e., the Action adapter, unless you want to develop a user interface configuration screen.

How do I create a new project?

1. Download the project source code
2. Pick a “name” for your project
3. Rename project alpha source code to your selected project name
4. Pick a color scheme for you manager application
5. Rearrange the manager menu’s for your application’s needs
6. Design your analytic – i.e., design your Action adapter

Hands on Training…

- If you have Visual Studio, we will work together step-by-step – otherwise just follow along and ask questions as needed.
- Preparation:
  - Survey who has Visual Studio 2012+
  - Deploy copies of Project Alpha code from thumb drive – or – download from web site
  - Launch Visual Studio – but do *not* open project yet

Choosing your Project Name

- First things first – you need to pick a name for your project. Here are a few things to keep in mind for the name:
  - Do not make the name too long – you can add a longer description where needed.
  - Do not use spaces your project name. The name needs to be an identifier that is safe for coding:
    - First character needs to be a letter
    - Name should consist of only letters and numbers
    - An underscore is allowed and counted as a letter
    - Upper case and lower case letters matter
    - For multiple words consider camel casing, like:

MyAnalyticsProject

Renaming your Code

- Make sure Project Alpha code is *not* open in Visual Studio. If it is, close is now.
- Run the “RenameProject” script from a command prompt using your project name as a parameter:

```
C:\ProjectAlpha\>RenameProject MyAnalytic
```

- Your Visual Studio project and all associated key source files will now be named according to your project name.
- If you want to change the name again, it is generally safer to start with a new clean copy of Project Alpha.

Opening your New Project

- Navigate to your project's "Main\Source" folder and open the solution file
- Visual Studio will now load your new project solution with projects for:
  - The base service
  - A console application
  - A manager application
  - A setup application – *requires WiX Toolset*:

    http://wix.codeplex.com/

  - SQL scripts for MySql, Oracle, SQL Server and SQLite
  - Configuration tools

Building your New Project

- From within the “Solution Explorer”, right click on the root of the solution hierarchy for your project and select “Rebuild Solution”
  - If necessary, right-click on the setup project and select “Unload Project” if WiX is not currently installed and prevents build
- When project build succeeds, navigate to the built solution folder, e.g.:

```
Main\Build\Output\Debug\Applications\MyAnalytic
```

Establish Initial Configuration

- To configure your new project for debugging, you will need to run the ***ConfigurationSetupUtility*** to create a new configuration database for your application
- Run this application and follow the prompts to create a new configuration database. SQLite will be the simplest option for initial debugging and testing. Use sample data if you do not have access to a source device

Update Source Configuration

- Now that you have an operational initial configuration, the default configuration in the application source code needs to be updated with this information to simplify development
- Close your application by right-clicking on its running icon in the task-bar (looks like a gear) and select “Exit”
- Open your application’s configuration file in the debug folder (e.g., MyAnalytic.exe.config) using a text editor

Update Source Configuration (cont'd)

- Copy the following key settings into the App.config of your service application and your manager application:
  - ConnectionString
  - DataProviderString
  - NodeID
- These three configuration settings define the needed information required to connect the current configuration database

Debug your Project in Visual Studio

- With an established configuration, you can now debug your project from within Visual Studio. This process will be essential for developing your analytic
- To start a debugging session, right-click on the service application and select “Debug / Start new instance”
- Your service project and console will now be started – placing a “Stop” in the code will now pause all running threads so you can step through code

Debugging the Manager

- To debug the manager application, right-click on the manager application and select “Debug / Start new instance”
- You can simultaneously debug both the service and the manager at the same time
- SQLite Note:
  - The SQLite dependency files are not automatically copied to the manager’s debug output folder, it may be necessary to copy “System.Data.SQLite.dll” and “System.Data.SQLite.Linq.dll” to the manager’s debug output folder before a successful run

Changing the Manager Color Scheme

- Navigate within your project's source folder to:

  `Main\Source\Documentation\Manager Colors\`

- View the sample image files here and choose a desired color scheme.
- The "backgrounds" and "foregrounds" text files in this folder define the needed script for the desired color scheme.
- Apply the desired color scheme settings to the manager source file "MainWindow.xaml", specifically, apply selected:
  - Background to MainBackgroundBrush linear gradient
  - Foreground to MenuBackgroundBrush linear gradient

Adding a Custom Analytic

- Right-click on the “Applications” folder in the solution and select “Add / New Project…”
- Select a new “Class Library” project and give the project a name – this will become the Action adapter for your analytic, e.g., MyAnalyticLib
- Add a reference to the following files that can be found in the GSF dependencies:
  - GSF.Core.dll
  - GFS.TimeSeries.dll
- Not inherit primary class from ActionAdapterBase, hint: add “using GSF.TimerSeries.Adapters;” to class header

Adding a Custom Analytic (cont'd)

- Right-click on "ActionAdapterBase" and select "Implement Abstract Class"
- Update the code as follows:

```
using System.ComponentModel;
using GSF.TimeSeries.Adapters;

namespace MyAnalyticLib
{
    [Description("MyAnalytic: This is my new analytic")]
    public class Class1 : ActionAdapterBase
    {
        public override bool SupportsTemporalProcessing
        {
            get
            {
                return false;
            }
        }

        protected override void PublishFrame(GSF.TimeSeries.IFrame frame, int index)
        {
            if (index == 0)
                OnStatusMessage("Received {0} frames so far...", PublishedFrames);
        }
    }
}
```