Catalyst WPF Overview

Where to put source language resources:

The source language is the language in which a piece of software has been developed. Resources in this language can be placed in a main assembly, which will then contain code and resources, or in a satellite, the satellite will then contain just resources and the main assembly will contain just code.

Microsoft recommends, if you intend to use their Locbaml solution for localising a WPF project, that source language resources be placed in a satellite assembly. However, when this satellite is inserted into Catalyst, we cannot produce a WYSIWYG view of the resources, so our recommendation is that the source language resources be stored in a main assembly. This is the default setting in the development environment. However, the following should be added in AssemblyInfo.cs, it explicitly states the location of the source language resources:

```
[assembly: NeutralResourcesLanguage("en-IE",
UltimateResourceFallbackLocation.MainAssembly)]
```

String Tables:

If you are writing a purely XAML GUI application, we recommend you don't use a ResX string table to store UI strings. Leave them all in XAML GUI declaration or alternatively a use resource dictionary. A resource dictionary is a XAML construct; however it does not need to store any UI, just a list of strings with IDs, wrapped in some XAML mark-up.
(They can also store style information, themes, binary resources etcetera, but for simplicity, each type of resource can be factored into a different resource dictionary and these can be merged later if required.)
The resource dictionary can be stored at application level and referenced throughout the WPF app. There is a lot of inbuilt support for this mechanism and it's very flexible when setup correctly, more so than ResX files.

File Preparation:

XAML (Extensible Application Markup Language; pronounced "zammel") is a declarative XML-based language that defines rich GUIs. A typical WPF project will consist of a mixture of code files and XAML files.

Catalyst requires that all UI elements in XAML have a unique identifier (UId) and will alert the user in the Results window if they do not appear to be present. UIds are used to keep track of changes to files and to identify items that must be translated. It's essential that they are present as Microsoft has not provided a mechanism to manually parse resources in assemblies, we are tied to their APIs and these in turn rely on UIds being present. To add UIds to XAML files in a project, the MsBuild tool must be run on the project file:

Msbuild /t:updateuid [Project Name].csproj

Then, to check that there are no missing or duplicate UIds run:

Msbuild /t:checkuid [Project Name].csproj

The project should now be recompiled. XAML is compiled by the development environment into BAML. BAML is then embedded in assemblies and this is what ships to customers.

Testing the satellite:

Having inserted our main assembly, with embedded UIds, into Catalyst, translated the resources into French, and then extracted a French satellite, we would like to see our WPF Application using the resources from our French satellite. We need make sure we have the following line in the constructor for the application:

```
Thread.CurrentThread.CurrentUICulture =
Thread.CurrentThread.CurrentCulture;
```

Place the satellite in an appropriately named (fr-FR) subfolder, under the folder our main assembly will run from. Now, change the operating system locale to French in the Windows regional options dialog. Run the application and it should appear in French.

Microsoft Approach:

The following describes how to follow Microsoft's recommendation and places language neutral resources in a satellite assembly:

Add a <UICulture>en-IE</UICulture> entry to the project's .csproj file . This automatically results in the language neutral resources being placed in an en-IE satellite . If you do this then you must also uncomment the following line in the project's AssemblyInfo.cs file:

[assembly: NeutralResourcesLanguage("en-IE", UltimateResourceFallbackLocation . Satellite)]

The culture in the above entry needs to match that specified by <UICulture> entry in the .csproj file . 'UltimateResourceFallbackLocation' should be set to 'Satellite', otherwise no resources will be found at runtime.

These language neutral satellites can be inserted into Catalyst. It will display all strings available for localisation but WYSIWYG will not be available.

- Assemblies containing a mixture of resource formats:
  In Visual Studio, when developing a WPF application, the user can choose to add a 'Resources File'. This adds a ResX file, essentially a string table, to the project and this will get compiled either to the main assembly or to a language satellite, depending on the project settings and how the ResX has been named within the project. Support for this was introduced in Catalyst 8 SP1. The resultant assembly will contain compiled XAML (.g.resources section containing multiple BAML streams) and compiled ResX (.resource section that contains name-value pairs).

Microsoft has a document on best practice for WPF L10T & G11N here:
http://blogs.msdn.com/wpfsdk/archive/2006/07/06/658392.aspx