Setup in Visual Studio to Build BrentDFIRAPP Project

Data Flow Intermediate Representation (DFIR) can provide users a graphical view of a source GVI or GMRD, which can represent all the VI constructs. Meanwhile, National Instruments offers their users an API, which equip the ability to read-only access to the LabVIEW Communications intermediate representation of VIs. The API is as a part of a special build of LabVIEW. Once the API is activated, the users can use it to build C# applications in Visual Studio platform. For a full understanding of API features, the users are encouraged to use Visual Studio with ReSharper to explore the API. For any class, the interface definition, inherited classes, and base classes can be viewed using the right-click context menu's Navigate section.

The establishment of DFIR C# application is in following steps:

- **Create A New C# Visual Studio Project**

When the users create a New Project, they can choose to establish the Class Library project as the project type according to "Installed - Templates - Visual C#". This operation will help the users create a C# .dll file, that can be connected with LabVIEW Communications actually.
And the users can customize the name and the solution name of the class by themselves. In addition, they can add multiple projects to the solution. In my research, I have built two C# projects: DotStringPrinter.cs and XMLGenerator.cs. Correspondingly, every project will match their own .dll library file. And each project can instantiate multiple buttons in the LabVIEW Communications' s UI.

- **Add A Class to the DFIR Project**

This class is used to represent one button in the LabVIEW GUI. The class is renamed to DotStringPrinter and XMLGenerator, a more sensible name, and then is refactored its name by ReSharper. Once this update has been done, other references in this solution will also be updated automatically.

- **Configure the DFIR Project to Export a LabVIEW GUI Button**

When the users need export a LabVIEW GUI Button, because the class must implement an object of *NationalInstruments.LabVIEW.DfirExport.IDfirRootAcceptor* interface, they need use the .NET Managed Extensibility Framework and some National Instruments components to achieve it.
Hence, the users need to add the following three lines on the top of the class definition in their C# project:

```
[Export(typeof(IDfirRootAcceptor))]
[PartMetadata("ExportIdentifier", "ProductLevel:Base")]
{PartCreationPolicy(CreationPolicy.NonShared)]
```

Furthermore, when the users need to find their class in the .dll file, they can add the following using directives to their class to make it possible for LabVIEW:

Eventually, the users need utilize the following sentences to replace the code in the "AssemblyInfo.cs" file in the Properties folder of the C# project:

Therefore, the users can edit the declaration of the class for the DotStringPrinter and XMLGenerator C# projects to make the IDfirRootAcceptor interface implemented:

```
using NationalInstruments.LabVIEW.DfirExport;
using NationalInstruments.LabVIEW.DfirExport.Interfaces.IDfirBase;
using NationalInstruments.Composition; ;
[assembly: ParticipatesInComposition]
public class DotStringPrinter : IDfirRootAcceptor
public class XmlGenerator : IDfirRootAcceptor
```

- **Add API Reference to the DFIR Project**

In the Solution Explorer window, the users can right click on Reference and Add Reference. In the browse section of Reference Manager, select the following two files:

NationalInstruments.LabVIEW.DfirExport.dll
NationalInstruments.Composition.dll files and select them.

- **Add Relevant System References to the DFIR Project**

Similarly, in the Framework, Assemblies of Reference Manager, the users need navigate and add the following system references:

System.CompositionModel.Composition
PresentationCore
WindowsBase
System.Xaml

- **Add a Name for the Button**

If the users want the button to display its name in the LabVIEW Communications GUI, they can make the *get* accessor of the *ButtonName* property return a string that contains the name. In my project, I designate the ButtonName returns *Print DotString* and *XMLGenerator*.

```
public string ButtonName
{
    get { return "Print DotString"; }
}
```

```
public string ButtonName
{
    get { return "XmlGenerator"; }
}
```

- **Add an Image for the Button**

The users can also add a .png image for their button in the LabVIEW Communications GUI. The image can be either the stock image or customized image. For the customized image, it should be 32x32 pixels without any transparency effects. In my research, I assign the DotStringPrinter and XMLGenerator C# projects to make the *get* accessor return null, which means that both of them use the default image.

```
public BitmapImage ButtonImage
{
    get
    {
        // Use the default image
        return null;
    }
}
```

- **Add the Button Action to the DFIR Project**

If the users want to add any action for LabVIEW to generate, they can add their code to the *AcceptDfirRoot* method. When the button is clicked in the LabVIEW Communications GUI, the code in the method will be executed and become the entry point of the C# application correspondingly. The class object will be instantiated when the VI is loaded by LabVIEW.
For my project, when I click the Print DotString Button, it will produce the .dot file to indicate the connections of the nodes in a graph. When I click the XMLGenerator Button, it will produce XML file to display the information of every node and their connections.