LabVIEW Plugins In InstrumentStudio

Registering a Plugin

In order to register a plugin, put a .gplugindata into the Addons directory under the InstrumentStudio installation directory. The .gplugindata file is an XML file that tells InstrumentStudio the properties of the plugin and how to find the code that implements it.

.gplugindata format

The .gplugindata file is an XML file. The easiest way to understand it is to look at an example.

```
<?xml version="1.0"?>
<GPluginMetadata xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.ni.com/InstrumentStudio/GPluginMetadata">
    <Plugins>
        <PluginData
            UniqueName="National Instruments|Example G Plugin"
            DisplayName="Example InstrumentStudio 2021"
            PanelType="Examples"
            GroupName="NI Example Plugins"
            VIPath="NI Example G Plugin.vi"
            SupportedPanelSizes="LargeAndSmall"
            ApplicationContext="Unique" />
    </Plugins>
</GPluginMetadata>
```

- The `GPluginMetadata` element is the root element containing all of the data to be registered with InstrumentStudio. You must specify the namespace for the schema as shown in the example above.
- The `Plugins` element contains the list of plugins
- Each `PluginData` element registers a single plugin and lists its properties. Attributes of the plugin are expressed as attributes of the `PluginData` element. The attributes recognized are:
    - UniqueName (required) - a unique string identifying this plugin. It should be globally unique across all registered plugins. It is recommended that you use a Guid or some other globally unique identifier which may include the company name to make it unique
    - DisplayName (required) - a name that will be displayed to the user for the plugin in all places in InstrumentStudio
    - SupportedPanelSizes (required) - set this to "Large", "Small", or "LargeAndSmall" to indicate which size layouts this plugin supports.
    - GroupName - The name of a group which will appear in the Edit Layout dialog. All plugins with this group name will be shown together in the dialog
    - PanelType - A string to categorize this plugin. It can be used to filter plugins (show only plugins of this type) in the Edit Layout dialog
    - VIPath (required) - a path to the top-level VI to load for the plugin. A relative path can be specified, in which case the path is considered to be relative to the gplugindata file's location. If

culture specific gplugindata files are installed underneath culture specific folders, the relative path is considered to be relative to the folder above the culture specific folder.
  - ProjectPath (required if ApplicationContext is set to "Project") - a path to the LabVIEW project in which to load the top-level VI. This setting only applies when ApplicationContext is set to "Project". A relative path can be specified, in which case the path is considered to be relative to the gplugindata file's location. If culture specific gplugindata files are installed underneath culture specific folders, the relative path is considered to be relative to the folder above the culture specific folder.
  - DebuggingEnabled (defaults to false) - Set this to true to make the application context of the plugin visible to LabVIEW as a debuggable context.
  - ReentrantExecutionEnabled (defaults to true) - Set this to false to better support debugging. Currently in LabVIEW 2020 and earlier there are some issues when debugging a VI that is running reentrantly that might be addressed in a future version of LabVIEW that you can use this setting to workaround, as follows:
    - The first issue is that there is no way to get at the reentrant VI from the debug session after it starts running. This generally means you have to save a breakpoint in the source VI before the clone is created and starts running.
    - However, the saved breakpoint in the clone also isn't being honored and the clone never breaks.
    - Similarly, if you set a breakpoint in a non-reentrant VI called by the clone and then try to step out/up into the reentrant VI, it just steps over and effectively does a resume running command.
  - ApplicationContext (required) - This is an enum value that can be either "Project", "Global", or "Unique"
    - Project - This allows you to share state between multiple plugins, or multiple instances of a plugin implemented within the same lvproj. If you use this option, you also need to specify the ProjectPath attribute.
    - Global - This allows you to share state between plugins globally, but there is a greater risk of conflict with plugins from other developers.
    - Unique - A unique context will be created for each instance of the plugin. This avoids the risk of the state of your VI conflicting with other plugins or instance of your plugin, but it also makes sharing data between plugins and plugin instances more difficult.

Relative Versus Absolute Paths

While there might be some use cases for absolute paths, such as enabling debugging of loose VIs without having to build a source distribution or PPL for it to find all of its dependencies by setting an absolute path to an installed LabVIEW's vilib folder under the AdditionalVISearchPaths setting, in general you should never ship a gplugindata file with absolute paths because that makes it harder to make a plugin that can work at any possible install location for InstrumentStudio.

Adding <vilib>, <userlib>, and <instrlib>

Our recommendation is to create lvlibp files or source distributions when distributing your plugins to end users. However, in order to support using loose, unpackaged VIs during development that can still reference VIs in the LabVIEW IDE's search paths, there is an option on the Preferences dialog under the Plugins category to automatically include the search paths for these folders if you have the correct version of the LabVIEW IDE installed. Note that G plugins always use the latest version of the backwards compatible LabVIEW runtime on the machine, so you will need to have the corresponding version of LabVIEW on your machine to debug VIs and to support this option.

Troubleshooting

If your plugin doesn't appear in the Edit Layout dialog, ensure the .gplugindata file is under the Addons directory of InstrumentStudio. If there is an error reading this file, InstrumentStudio will display a message box with details about the error.