Instructions to SDK Vendors for Generating Pulsar Metadata

David Dubrow
(Nokia, Inc.)
Sequoyah Project
April 27, 2010

Modifying the p2 metadata

The P2 metadata for the installable units (IUs) representing each SDK must pass a query for feature groups. This boils down to two things:

1. The id of the IU must end with ".feature.group"
2. The IU must contain the group property with a value of "true"

The new metadata generator will ensure this is true for IUs it generates, but existing p2 content.xml files may be modified by hand if this is easier for some SDK vendors.

To modify by hand, it should be noted that the id of the IU shows up in two places in each <unit> element in the content.xml file:

1. As the *id* attribute of the element
2. As the *name* attribute of the *provided* sub-element.

The group property may be added to the *properties* element of the IU (copy and paste the following):

```
<property name='org.eclipse.equinox.p2.type.group' value='true'/>
```

The discovery metadata

The discovery metadata is provided as extensions in a plug-in. This plug-in is not provisioned via p2 to users with the SDK installation. It is only a vehicle for providing metadata to the p2 discovery catalog viewer in the Pulsar SDK view.

The p2 discovery catalog viewer in the Pulsar SDK view is fed catalog item metadata and category metadata from two sources:

One is a discovery directory xml file with URL entries to the plug-ins with the discovery extensions. The file is managed by the Sequoyah project and can be modified with URL entries to plug-in jar files at any time.

The other is source is local (for testing). Any plug-in containing these extensions in the current installation will also be used in the Pulsar SDK view. This allows testing of the discovery metadata while developing it. Pulsar will not actually contain any plug-ins with metadata when deployed. Using eclipse with Pulsar and the PDE will allow testing without deploying the plug-in with the discovery metadata.

One benefit of using extensions for the metadata is that the metadata can be generated using eclipse's Plug-in Development Environment (PDE). The plugin.xml editor can be used to edit the discovery extensions, and the plug-in can be exported via the "Export…/Deployable Plug-ins and Fragments" wizard.

Instructions for creating the discovery metadata plug-in

1. Create a new project using the "New Project…/Plug-in Project" wizard (the checkbox for the Java project with source and output folders can be unchecked since there will not be any Java code in the project).
2. The project will have a META-INF folder with a MANIFEST.MF file in it and a build.properties file.
3. Add the icons and screenshots to the project. I suggest creating a folder with the New/Folder wizard – and name it "Images" (the name doesn't matter). Each category can have a 48x48 icon, each catalog item (sdk) can have a 32x32 icon and screenshots can be provided for catalog items and must be 320x240. By adding them to the project in advance, they can be selected more easily when creating the extensions.
4. Open the build.properties file and it will open in the Build Configuration editor. Ensure the "Images" folder is part of the Binary Build (check its checkbox in the Binary Build tree) (SAVE)
5. Open the MANIFEST.MF file and it will open in the multi-tab PDE editor.
6. In the Overview tab, check the "This plug-in is a singleton" checkbox
7. In the Dependencies tab, add "org.eclipse.equinox.p2.discovery.compatibility" (SAVE)
8. In the Extensions tab, add an extension of type "org.eclipse.mylyn.discovery.core.connectorDiscovery" (this will actually generate a new plugin.xml file in the project that will contain the extension metadata).
9. Selecting the new extension, use the context menu to add connector categories and connector descriptor elements representing the categories and SDKs descriptors.

Category attributes

Categories (*connectorCategory*) require an *id*, a *name* and a short *description*.

The name and description will show up in the viewer and should be tested to provide the desired results (see Testing the discovery metadata below). A category must have a *group* sub element with an *id*. (I use the same id for both the category and the group with no adverse effects).

A category can optionally contain a 48x48 *icon* (also a sub element). The icon can be selected from the project using the "Browse…" button for the "*image48*" attribute.

Descriptor attributes

Descriptors (*connectorDescriptor*) representing the SDK require a *kind* attribute, which must be set to be exactly the word "task" (no quotes), a *name* that is visible in the viewer, a *provider* (company name) and a *license* (e.g., EPL 1.0).

The *siteUrl* attribute should be the URL to the p2 repository (where the content.xml and artifacts.xml exist). The *id* for each descriptor is the id of the IU for the SDK (the one that must end with ".feature.group" – see above). The *categoryId* is the id of the category and the *groupId* is the category's group's id. A short *description* can be added and an optional *platformFilter* attribute can be added to filter the catalog item by platform (e.g., osgi.os=macosx).

Descriptors can optionally contain a 32x32 *icon* (also a sub element). The icon can be selected from the project using the "Browse…" button for the "*image32*" attribute.
Finally, the descriptor should have an *overview* sub element. If one is added, an information icon is added to the catalog item in the viewer. Clicking the information invokes a popup that may contain any of: a summary (a long description – must fit in a 320x240 space in whatever font is used by the catalog viewer – this must be tested to ensure it looks ok), a *screenshot* (a 320x240 image) and a *url* to a web page with additional information. The popup is constructed based on which of these attributes exists in the overview element.

Testing the discovery metadata

From your eclipse environment that contains Pulsar, you can run (Ctrl-F11) and this will generate a launch configuration that uses the development environment as a target platform plus any plug-ins in the current workspace. In the running instance, open the Pulsar SDK view (or switch to the Pulsar perspective). Your categories and SDK descriptor catalog items should be visible in the Pulsar SDK view. This way, you can get your items to look as desired prior to deployment.

Note: Using this method, the SDK might not be installable. I could only install items when the discovery metadata plug-in was actually added to the installation (i.e., placing the deployed plug-in in the dropins folder and restarting the eclipse environment may work if you can't install while running from a launch configuration – see next section for deployment of your metadata plug-in).

Deploying the discovery metadata

From the "File/Export…" menu, select the "Deployable plug-ins and fragments" wizard and export your plug-in as a jar file. Place this on a server that does not require authentication (I don't believe a server that requires authentication can be used). Contact Sequoyah to add the URL to the deployed plug-in jar file to the Sequoyah discovery directory xml file.