

## Notification Extension Example

NEE - Introduction .....	1
NEE - Java API Workspace preparation .....	1
NEE - Creating project plugin .....	1
NEE - Deployment to Installed Polarion .....	1
NEE - Execution from Workspace .....	1
NEE - Configuration .....	1

See also main SDK page.

### NEE - Introduction

In this example we will show how to create custom notification target.

**what is possible to extend**

- **custom target** - extend configuration in Administration -> Notifications -> Targets

**what will be shown in the example**

1. how to create custom target - we will implement `custom-field-targets` target which will ensure that notifications will be sent to users with IDs found in certain custom field

### NEE - Java API Workspace preparation

See section *Workspace preparation*

### NEE - Creating project plugin

You can import already implemented example or read what steps are necessary to extend Polarion notification system.

#### NEE - Import of the example

Info: You must ensure that your plugin is compiled against your Polarion version. This example contains precompiled jar plugin. You can remove it before you start developing your plugin based on this example. The Eclipse ensure that new jar plugin will be created against your source code and Polarion version.

To import workflow project example to workspace, do these steps:

1. Select **File > Import...**
2. In the dialog that appears, select **Existing Project into Workspace** in **General** section and press **Next** button.
3. By pressing **Browse..** button, select the directory of examples (mostly in `C:\Polarion\polarion\SDK\examples\`). Submit it.
4. Select `com.polarion.example.notifications` and press **Finish**.

#### NEE - Extending Polarion notification system in own way

- Create new plug-in project. Fill Plug-In Properties and uncheck **Generate activator..**
- Create `META-INF` directory in `src` folder and `hivemodule.xml` file inside.
- In `hivemodule.xml` you can set one contribution point:
  1. `com.polarion.psvn.core.notifications.targets` to register new targetSee `hivemodule.xml` file included in example for syntax and more details.
- Open `MANIFEST.MF` and set `com.polarion.alm.tracker`, `com.polarion.platform.persistence`, `com.polarion.psvn.launcher` as a Required Plug-in in Dependencies card. As well, you should set at Build card the `src/` folder as the source folder that should be compiled into exported library.

```
### content of 'build.properties' file ###

source.. = src/
output.. = bin/
bin.includes = META-INF/, \
    .
```

- See how to manually set targets for email notifications in the documented example code.

### NEE - Deployment to Installed Polarion

See section *Deployment to Installed Polarion*

### NEE - Execution from Workspace

See section *Execution from Workspace*

### NEE - Configuration

After successful deployment of plug-in into Polarion, you can modify notification configuration to start using new event and target:

1. Select the Repository or project view. Go to Administration perspective, choose Targets in Notifications section, where you can add following code:

```
<workitem-commented>
  <custom-field-targets fieldId="notifiedUsers" />
</workitem-commented>
```

When you use `custom-field-targets` in `workitem-commented` then users which ids will be filled in custom field `notifiedUsers` (delimited by ',') will be notified when new comment will be added to particular workitem.

2. Definition of used custom field (Work Items > Custom Fields):

```
<field id="notifiedUsers" type="string" name="Notify users about WI update"
  description="" required="false" />
```

## Requirements

### Development Environments

- [Eclipse IDE for Java EE Developers](#) or any other Eclipse IDE with The Eclipse Plug-in Development Environment (Go to Help > Install New Software... > Install Eclipse Plug-in Development Environment > Restart Eclipse)
- [Java Platform, Standard Edition 8 Development Kit](#) or Java bundled with your Polarion installation

### Polarion Server

- Polarion Server with version 3.1.0 and higher

## Workspace Preparation

To start developing a Polarion Java API plug-in, you first need to perform following steps:

1. Start Eclipse, then select **Window > Preferences...**
2. In the dialog that appears, select **Plug-In Development > Target Platform**.
3. Click the **Add** button on the right.
4. Keep the **Nothing: Start with an empty target definition** option selected and click **Next**.

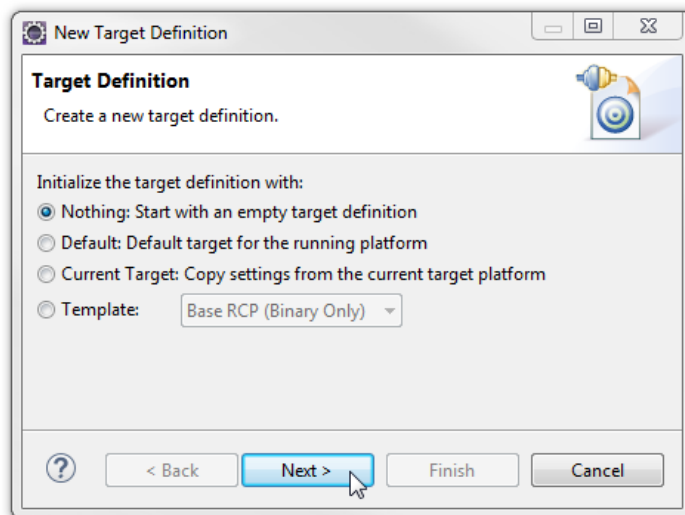

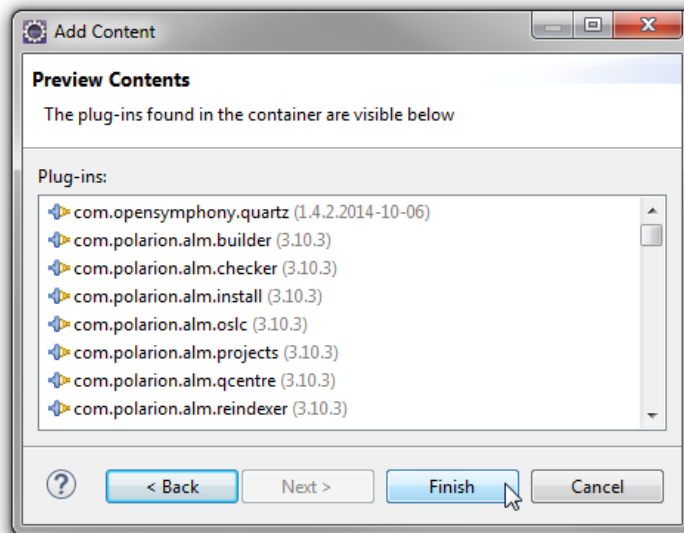


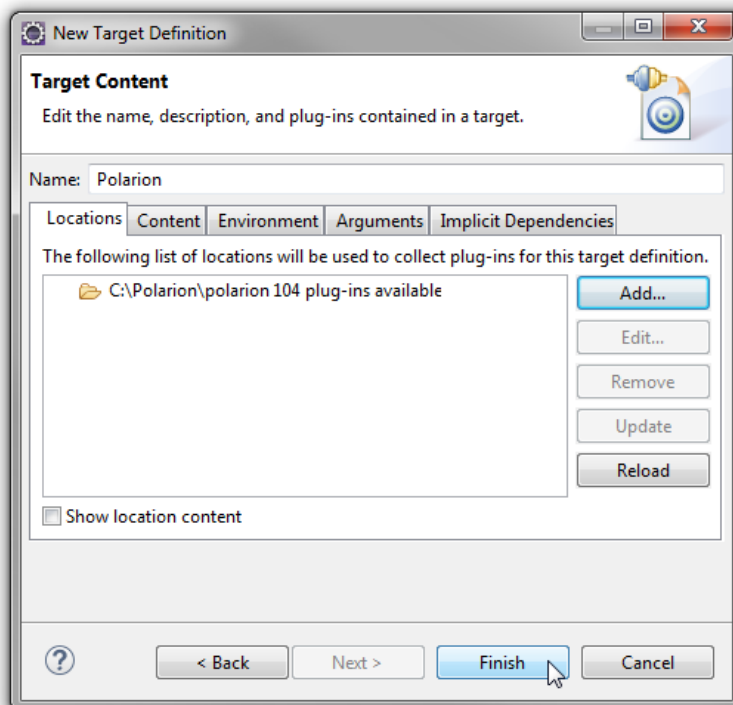
Figure WP-1: Starting with an Empty Target Definition

5. Enter a **Name** and click **Add**.
6. Select  **Directory** and click **Next**.
7. Click **Browse**, select the `C:\Polarion\polarion` folder and click **Next**.



**Figure WP-2:** Currently Installed Polarion Plug-ins

8. A list of currently installed Polarion plug-ins appears. Click **Finish**.



**Figure WP-2:** Confirm the Selected Path

9. The selected path and the number of discovered plug-ins available appear. Confirm that the path is correct and click **Finish**.

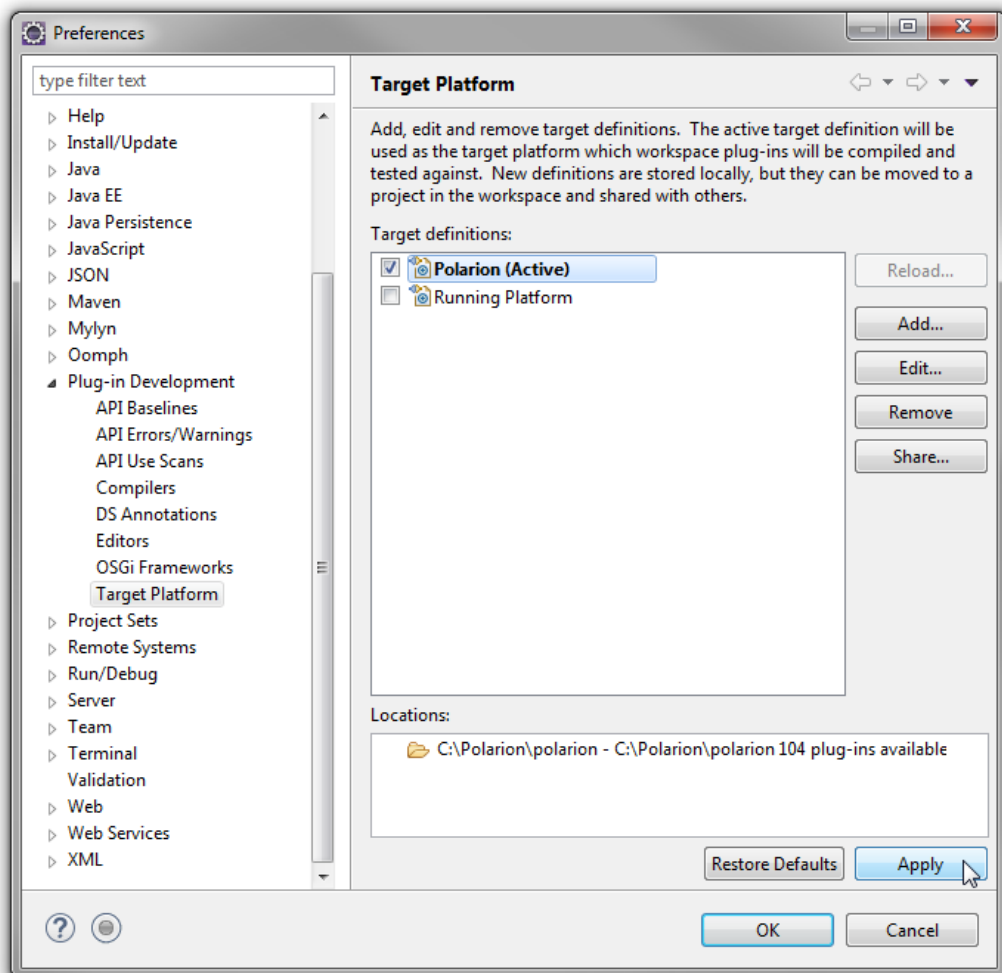


Figure WP-2: Select the Target Platform

10. Check the box beside the newly added path and click **Apply**.

## Deployment to Installed Polarion

You can deploy a plugin to Polarion in two ways. First you can export a project as **Deployable Plugins and Fragments**. The second way is described in the following section *Execution from Workspace*. To export the plug-in, perform these steps:

1. Select **File > Export...**
2. In the dialog that appears, select **Deployable Plugins and Fragments** in **Plug-in Development** section and click the **Next** button.
3. Mark your project (e.g. for **Servlet** example it will be `com.polarion.example.servlet`), and as the destination directory specify the `polarion` folder of your Polarion installation directory (usually in `C:\Polarion\polarion`)
4. At the **Options** card be sure, that **Package plug-ins as individual JAR archives** is unchecked. Click **Finish**.
5. Because this is a new polarion plug-in extension, you have to restart your Polarion server.

## Execution from Workspace

The second way to deploy the plug-in to Polarion is to launch Polarion directly from your Eclipse workspace. This method has the added advantage of debugging the code directly in Eclipse.

1. Select **Run > Open Debug Configurations...**
2. Create a new Eclipse application (double click on *Eclipse Application*)
3. You should set:

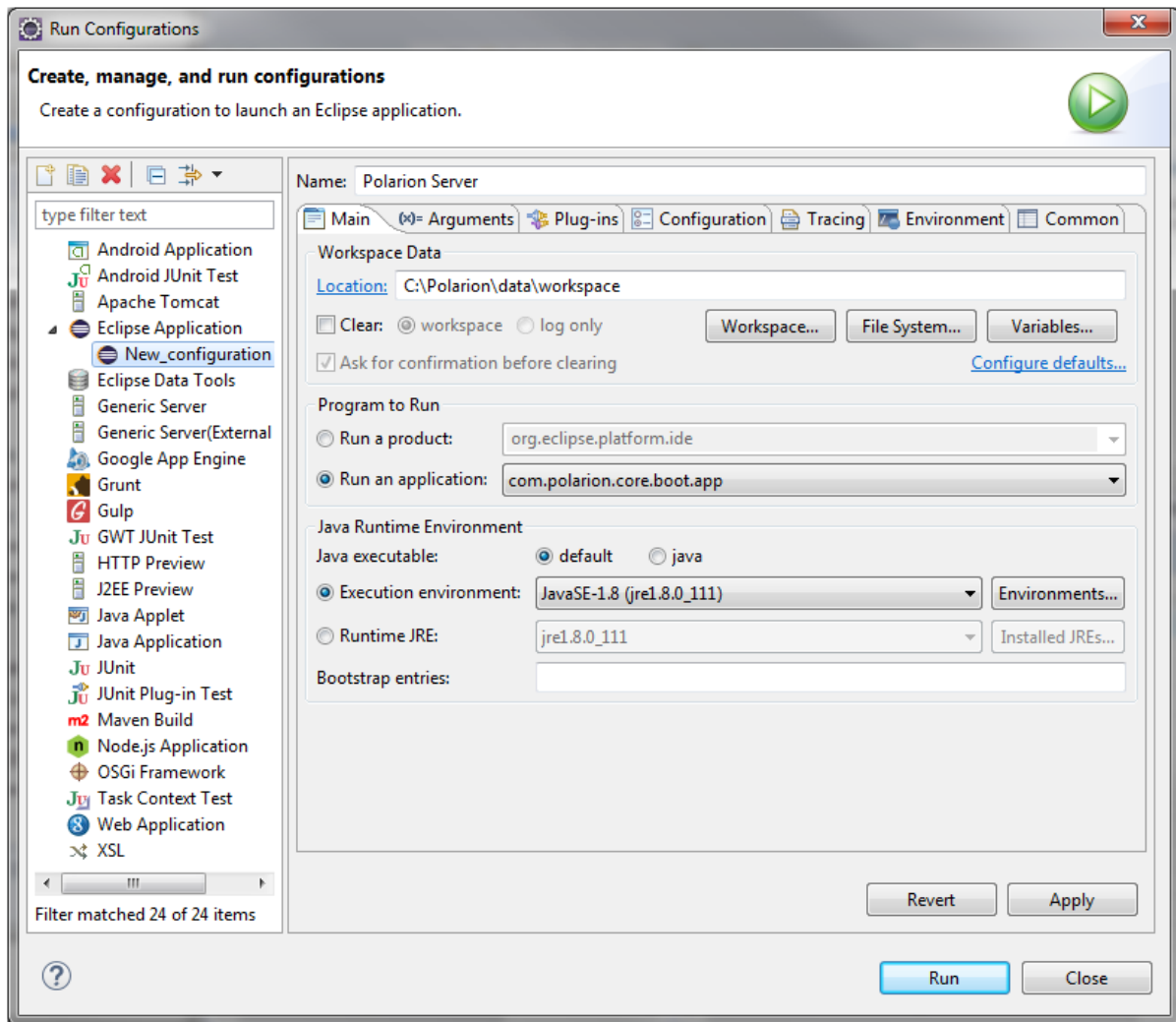


Figure WP-2: Debug - Main page

- **Name** to Polarion Server
  - **Workspace Data Location** to C:\Polarion\data\workspace (assuming that your Polarion is installed in C:\Polarion\)
  - **Run an application** to com.polarion.core.boot.app in the **Program to Run** section
  - Finally set your Runtime JRE
4. On the second, "**Program Arguments**" tab, set the following arguments:
- to **Program Arguments** section:

```
-os win32 -ws win32 -arch x86 -appId polarion.server
```

1.

- to **VM Arguments** section:

```
-Xms256m -Xmx640m -XX:MaxPermSize=128m -XX:PermSize=128m  
-Xbootclasspath/a:C:\Polarion\bundled\java\lib\tools.jar  
-Dcom.polarion.home=C:\Polarion\polarion
```

Of course, you have to change parameters to Polarion server according your installation.

Note that, when using Java 8 the arguments `XX:MaxPermSize` and `XX:PermSize` are obsolete and must not be used anymore.

1.

- You can check the settings with the following screenshot:

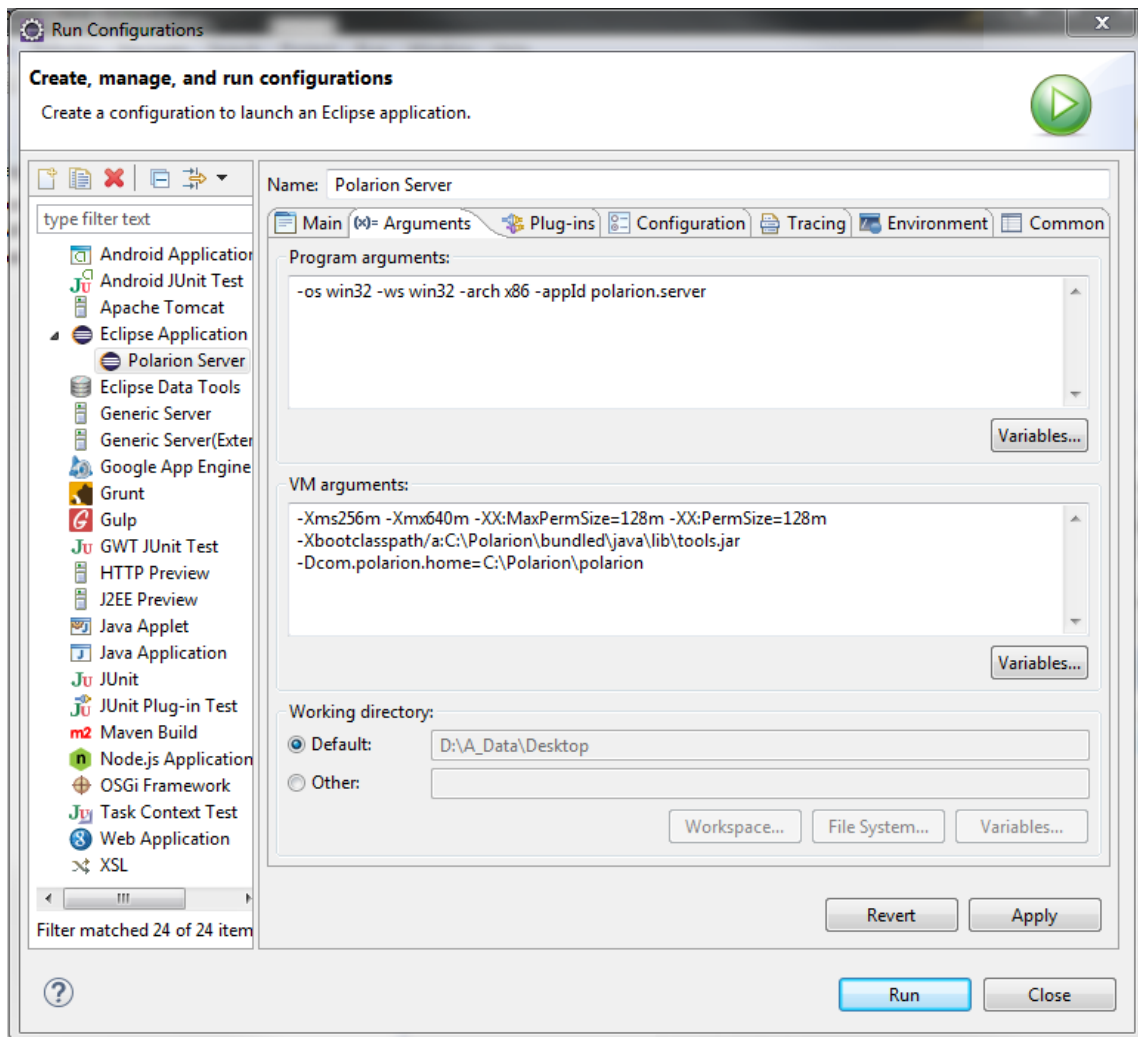


Figure WP-3: Debug - Arguments page

2. On the third "**Plugins**" tab, make sure, you have also selected "**Target Platform**" plugins.
3. Select all, and then click the **Validate Plug-ins** button. If there are some problems, uncheck the plugins which are in conflict.

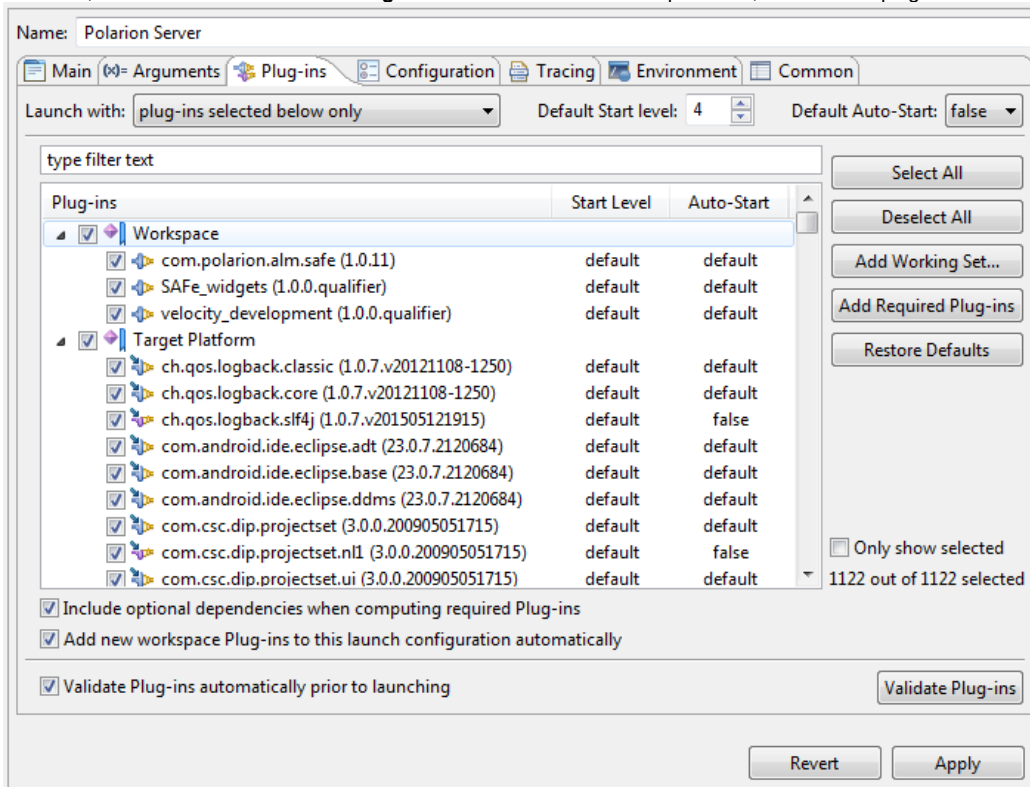


Figure WP-4: Debug - Plug-ins page

4. Other pages shouldn't be changed. Just click the **Debug** button, and go on with your new Polarion Server application.