

Job Example

JE - Introduction	1
JE - Java API Workspace preparation	1
JE - Creating project plugin	1
JE - Deployment to Installed Polarion	1
JE - Execution from Workspace	1
JE - Configuration	2

See also main SDK page.

JE - Introduction

The Job example is an implementation of a custom job unit. The implementation checks work items on due date and, if work items are delayed, it sends notification to assignee or to global email box. The example also covers building a parameterized job unit, which allows you to define parameters for a job and send announcements through the `IAnnouncerService` API class.

This example also shows you another part of Polarion, which allows you to create an extension for the scheduler system. The scheduler executes jobs periodically and it can compute some statistics based on work items.

JE - What is Scheduler and Job?

Scheduler is organizer for jobs. You can edit your jobs through the scheduler for periodic execution or monitor your jobs. Scheduler is inspired by Cron scheduler and you can easily maintain executions with Cron expressions. It allows you to set executions every five minutes, every hour or however you like.

Jobs are the implementation unit for the Scheduler and they are periodically executed by the Scheduler. Action of a job can vary widely - from some trivial calculation based on work items to sending periodic build analysis reports.

JE - Overdue job: Check for overdue work items

The typical usage of work items is that it allows you to specify the due date for resolution. If you have many work items in tracker, it's quite complicated to inform every user about his delay. This task can be easily realized with Job. You can extend Schedule for your own job that can periodically check every work item if it's resolved to due date or not. And then if you find some delayed work items you can send notifications for every assignee of work item.

JE - Java API Workspace preparation

See section *Workspace preparation*

JE - Creating project plugin

JE - 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.job` and press Finish.

JE - Hints to develop your own plug-in

- First, you have to create new plug-in project. Fill Plug-In Properties and uncheck **Generate activator..**
- Afterwards, open `MANIFEST.MF` and set `com.polarion.alm.tracker`, `com.polarion.platform.jobs` 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/, \
```

JE - Deployment to Installed Polarion

See section *Deployment to Installed Polarion*

JE - Execution from Workspace

See section *Execution from Workspace*

JE - Configuration

After successful deployment of plug-in into Polarion, you can start using new job in Scheduler. To check that deployment was successful do following steps:

1. Select the Repository view. Go to Administration perspective, choose the Scheduler, where you can add your new job.
2. Edit the global configuration for jobs and add these lines for the Overdue job example:
Job can be programmable with properties which will be injected into the job implementation by the Scheduler.

```
<job name="overdue.job" id="overdue.job" cronExpression="0 0 1 ? * MON-SAT"
  scope="project:playground">
  <query></query>
  <sort>~updated</sort>
  <notificationSender>polarion@example.com</notificationSender>
  <notificationSubjectPrefix>[Polarion]</notificationSubjectPrefix>
  <notificationRecipients>assignee</notificationRecipients>
  <planningConstraint>dueDate</planningConstraint>
  <allowedDelay>0</allowedDelay>
</job>
```

3. Then you can switch to the Projects perspective, choose Monitor and monitor your new job:

Scheduled Jobs			
Execute now			
<input type="checkbox"/>	Name	Scope	Cron Expression
<input type="checkbox"/>	Cleanup of Temporary Files	system	0 0 6 ? * *
<input type="checkbox"/>	Index Checker	system	0 0 23 ? * *
<input type="checkbox"/>	Index Refresher	system	0 0 0 ? * * 2099
<input type="checkbox"/>	All Calculations	system	0 0 0 ? * * 2099
<input type="checkbox"/>	Work Item Analysis	system	0 0 0 ? * *
<input type="checkbox"/>	Live Plan Chart Update	system	0 0 23 ? * *
<input type="checkbox"/>	Repository Analysis	system	0 0 1 ? * MON-SAT
<input type="checkbox"/>	Process Audit	system	0 0 3 ? * MON-SAT
<input type="checkbox"/>	Live CMMI	system	0 0 4 ? * MON-SAT
<input type="checkbox"/>	overdue.job	project:playground	0 0 1 ? * MON-SAT

Figure JE-1: Screenshot of monitor of the new job

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**.

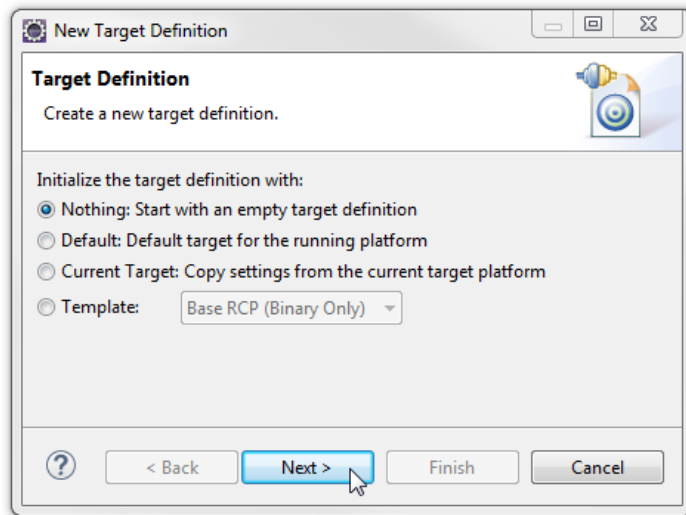



Figure WP-1: Starting with an Empty Target Definition

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**.

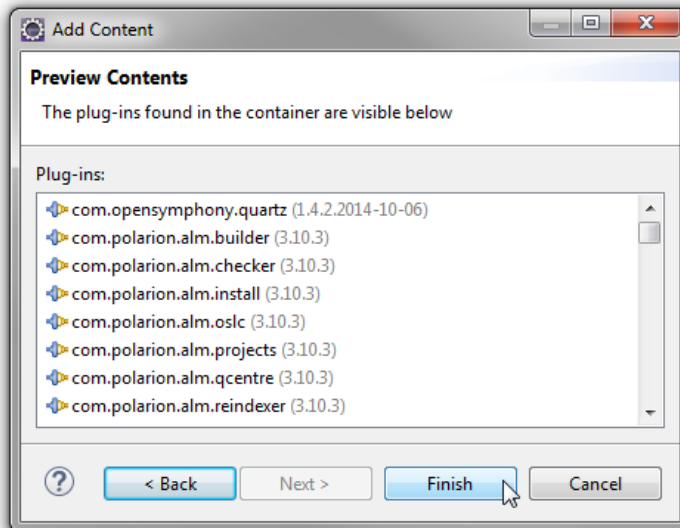


Figure WP-2: Currently Installed Polarion Plug-ins

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

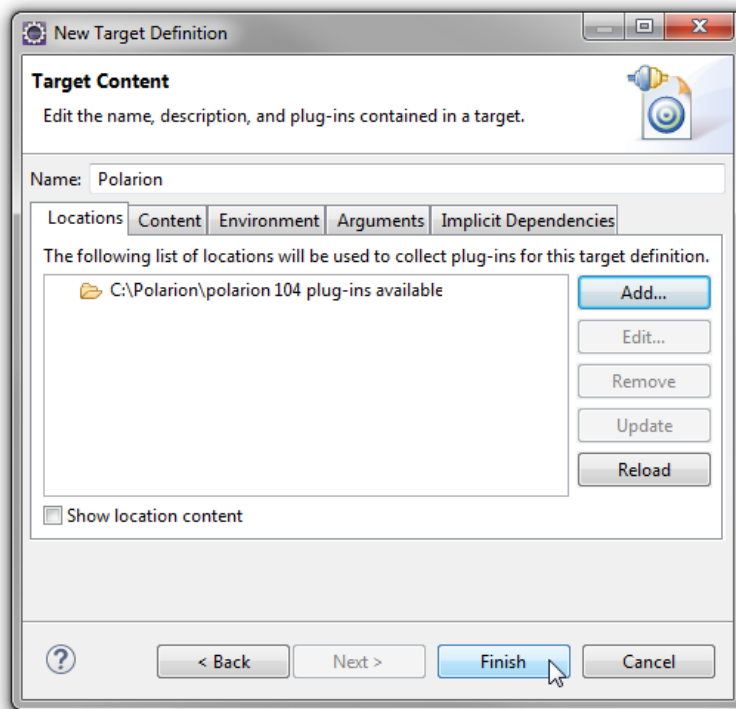


Figure WP-2: Confirm the Selected Path

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

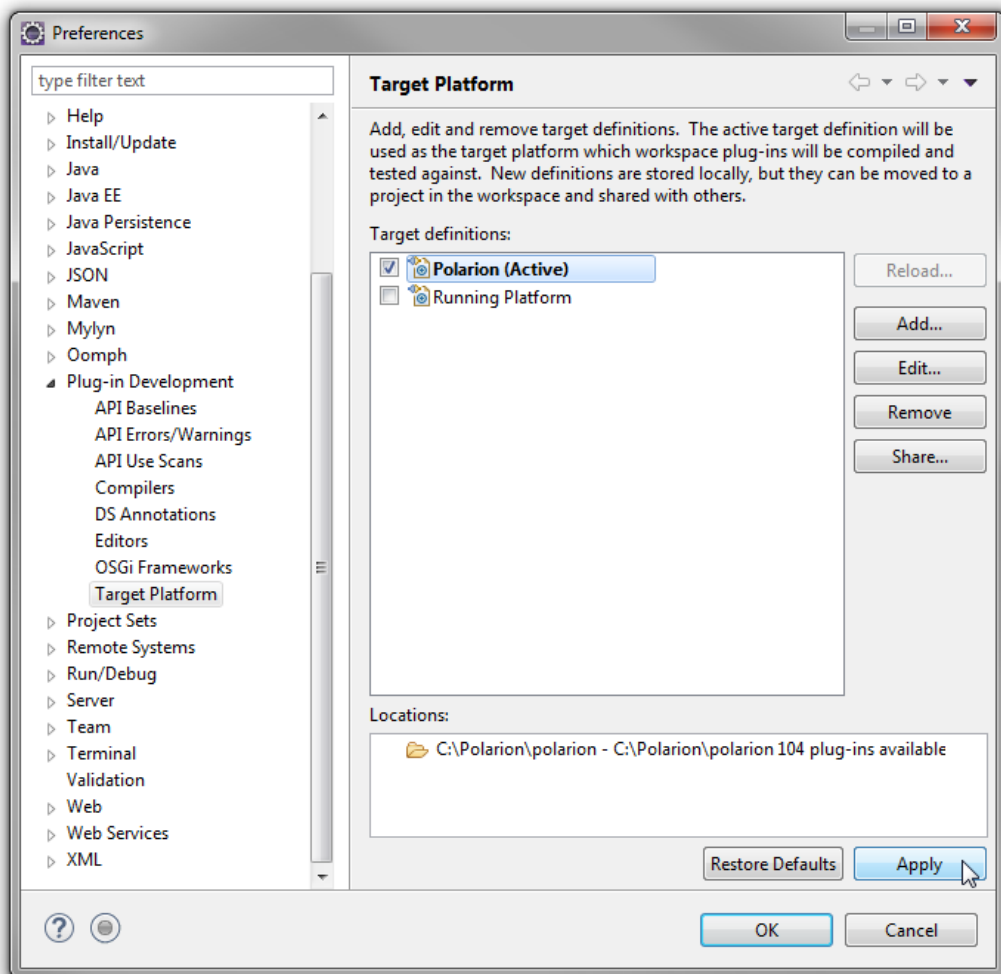


Figure WP-2: Select the Target Platform

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:

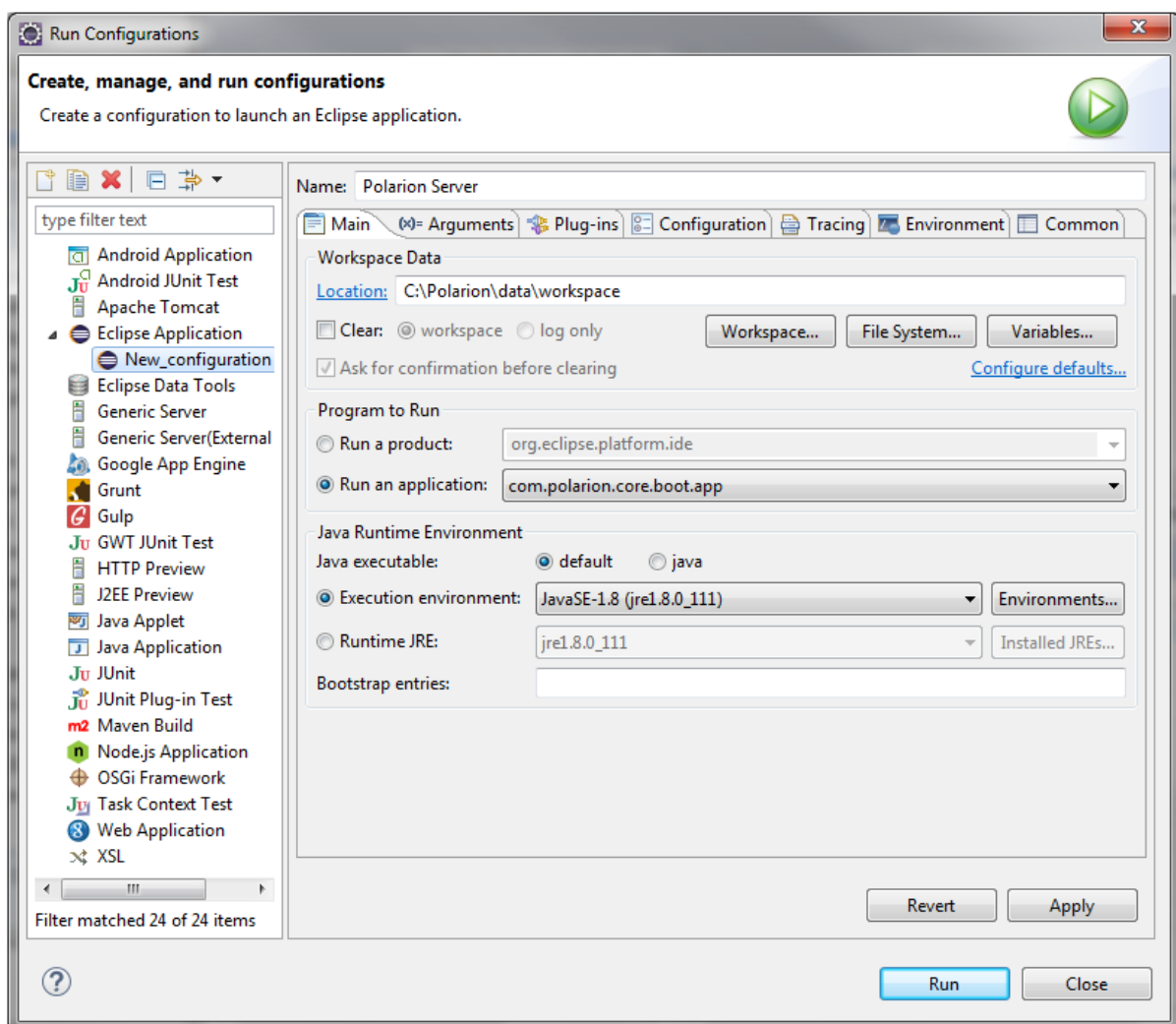


Figure WP-2: Debug - Main page

- **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:

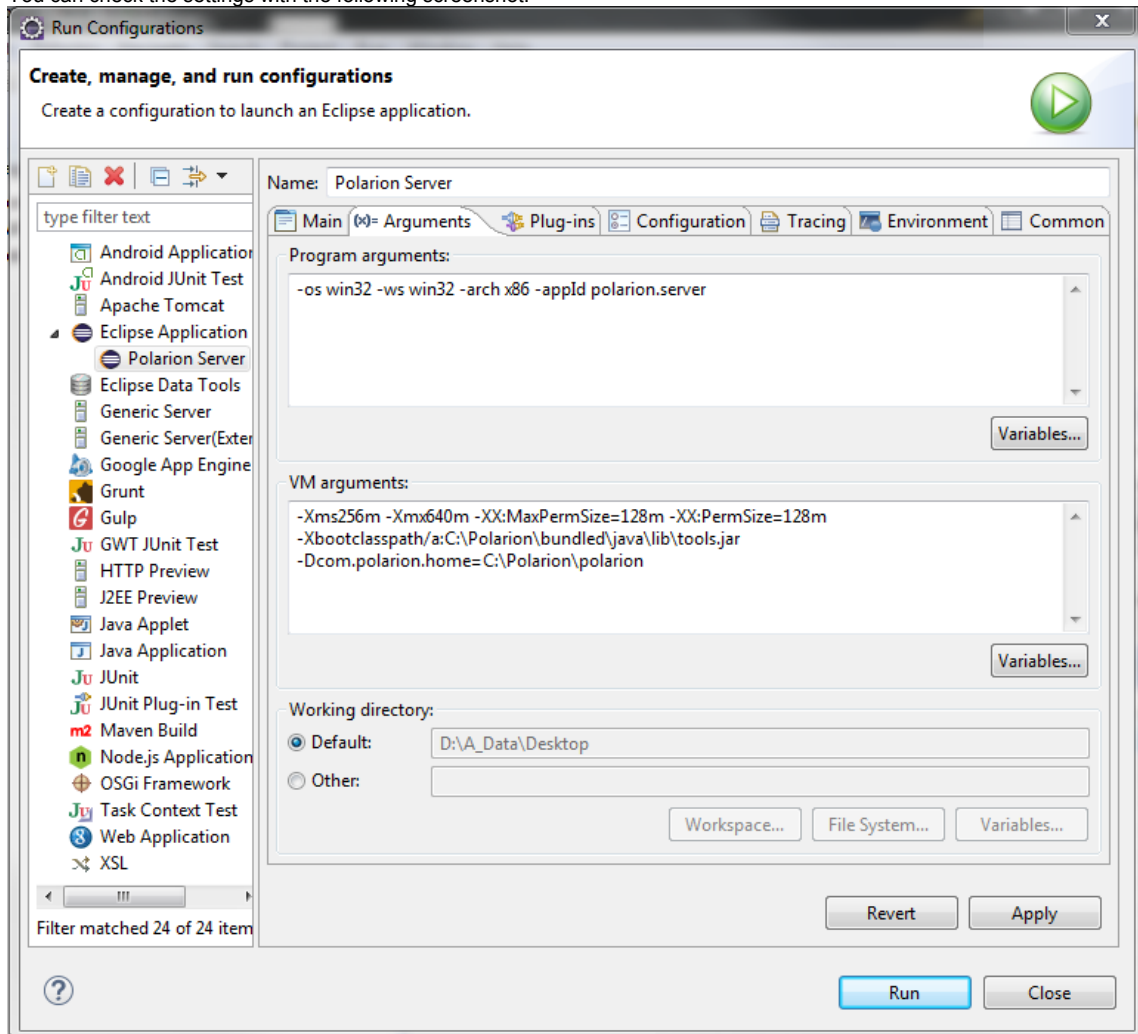


Figure WP-3: Debug - Arguments page

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.

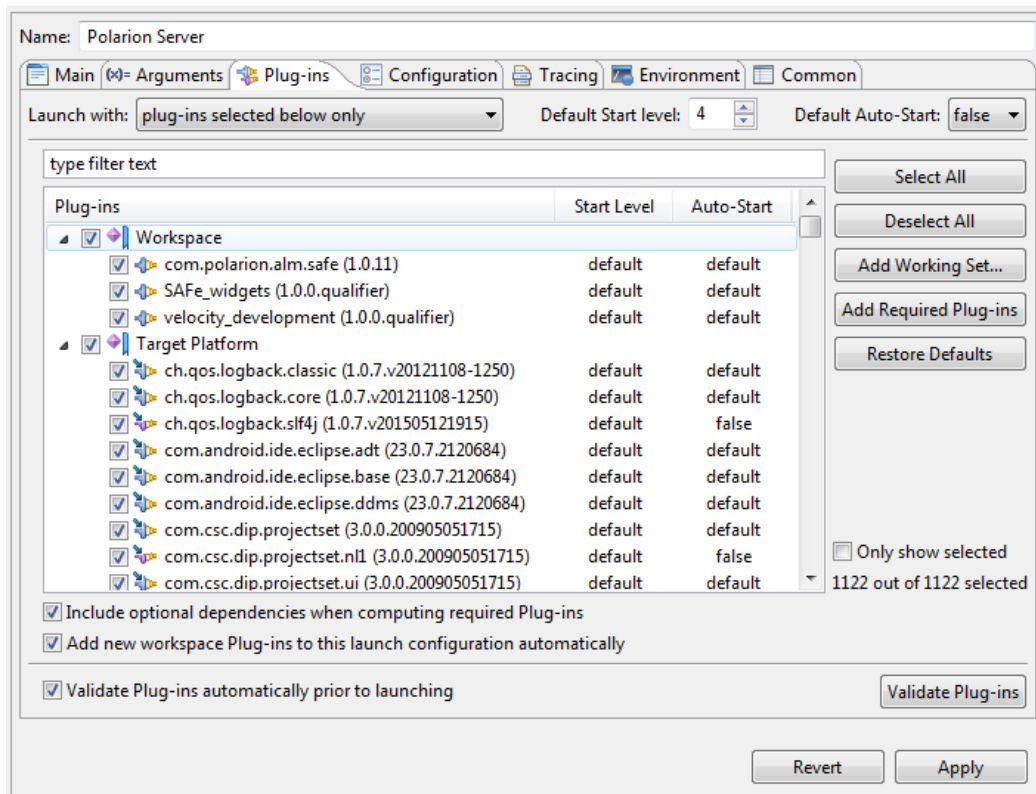


Figure WP-4: Debug - Plug-ins page

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