LLM Provider Example

See also main SDK page.

LLM - Introduction

This example demonstrates how to create a custom provider of Large Language Models (LLMs).
The example uses a locally running Ollama server and makes any model running on this server available for use in Polarion.

LLM - Java API Workspace preparation

See section *Workspace preparation*

LLM - Import of the example plugin

To import this example into your workspace:

1. Select **File** > **Import...**
2. In the dialog that appears, select **Existing Project into Workspace** in the **General** section and click **Next**.
3. Click **Browse...**, select the examples directory (usually in `C:\Polarion\polarion\SDK\examples`), and confirm.
4. Select `com.siemens.polarion.example.llmprovider` and click **Finish**.

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

LLM - Deployment to an Installed Polarion

See section *Deployment to an Installed Polarion*

LLM - Execution from Workspace

See section *Execution from Workspace*

LLM - Ollama server installation

- Download and install the Ollama app from https://ollama.com/ on the server where Polarion is running.
- After installation, no model is available by default, so pull one first. For example, run this command to pull the `llama3.2` model:

```
ollama pull llama3.2
```

LLM - Configuration of Copilot and REST API in Polarion

To use LLMs in Polarion, you must enable the Copilot feature. To try out the REST API and Swagger UI, you must also enable them.

- Specific permissions and an add-on license are required.
- The suggested settings are summarized below.
- For additional details, see Polarion Help and the REST API User Guide for Polarion.

Set the following system properties: (In the `polarion.properties` file.)

```
com.siemens.polarion.copilot.enabled=true
com.siemens.polarion.rest.enabled=true
com.siemens.polarion.rest.swaggerUi.enabled=true
```

For any users that will try out the example below:

- Add the Copilot add-on license.
- Grant repository-level permission "`Polarion Copilot > Permission to USE`".
- Grant repository-level permission "`Polarion Copilot > Permission to USE API`".
- Grant repository-level permission "`REST API Endpoints > Permission to POST`".

LLM - Configuration of Ollama model in Polarion

After successfully deploying the plugin to Polarion, you can configure an LLM to use the new provider "ollama".

In the repository-level `Administration > Configuration Properties`, add the following properties:

```
copilot.model.basic.provider=ollama
copilot.model.basic.serviceUrl=http://localhost:11434
copilot.model.basic.model=llama3.2
copilot.model.basic.temperature=0.5
```

Now you can use this model for your use cases via the Java or REST APIs.
If you want to use it for Polarion Copilot use cases (Content Validation, etc.), you must set the model as the default:

```
copilot.default.model=basic
```

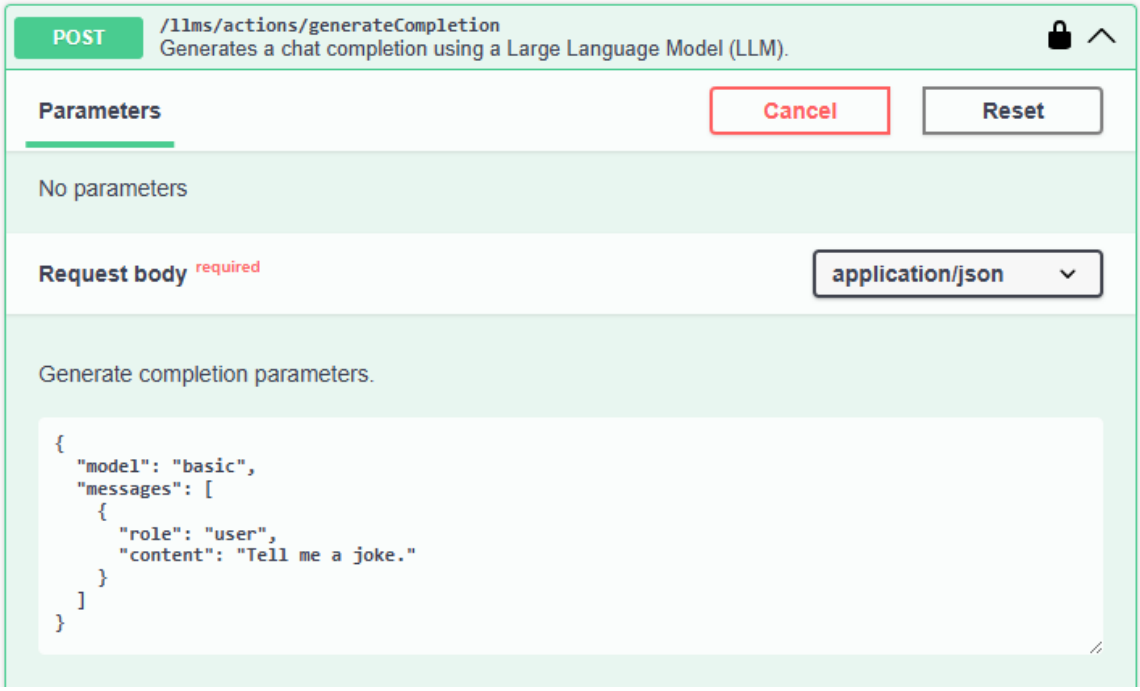
You can simply try out this model using the Swagger UI by accessing `https://[Your_Polarion_server]/polarion/rest/v1` in your browser.

(See the "REST API User Guide for Polarion" for instructions on how to authenticate.)

For example, use the following request body for the REST API endpoint `POST /llms/actions/generateCompletion`:

```
{
 "model": "basic",
 "messages": [
   {
     "role": "user",
     "content": "Tell me a joke."
   }
 ]
}
```

This is how this request looks in Swagger UI:



Requirements

Development Environments

- Eclipse IDE for Enterprise Java 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)
- Eclipse Temurin™ 21 (LTS) by Adoptium for building and running your code.

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**.

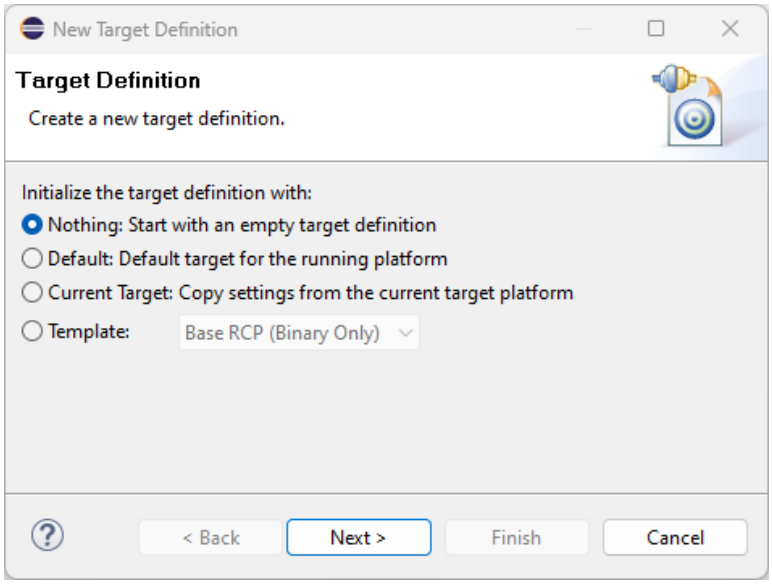


Figure WP-1: Starting with an Empty Target Definition

5. Enter a **Name** and click **Add**.
6. Select **Directory** and click **Next**.
7. Click **Browse** and select the `C:\Polarion\polarion` folder *(Windows)* or `/opt/polarion/polarion` *(Linux)*. (

One level above the *plugins* folder.)

8. Click **Next**.

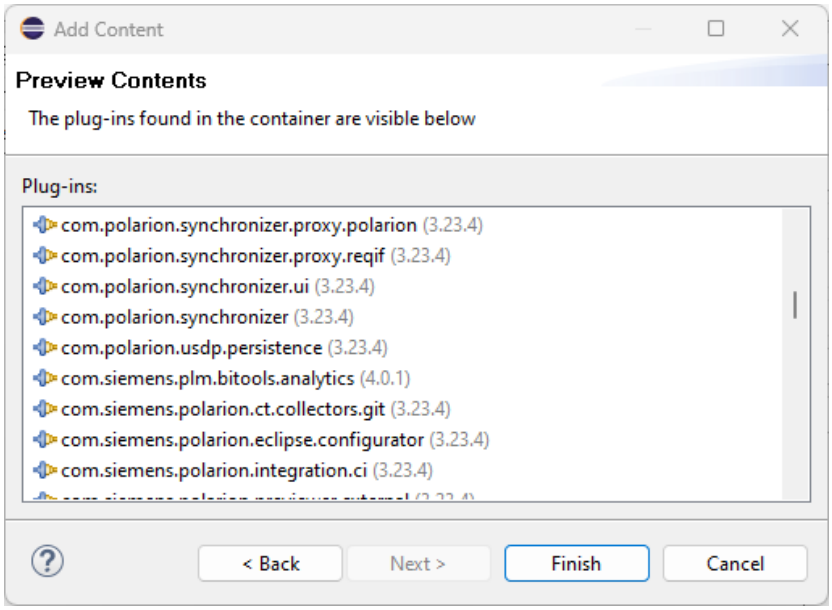


Figure WP-2: Currently Installed Polarion Plug-ins

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

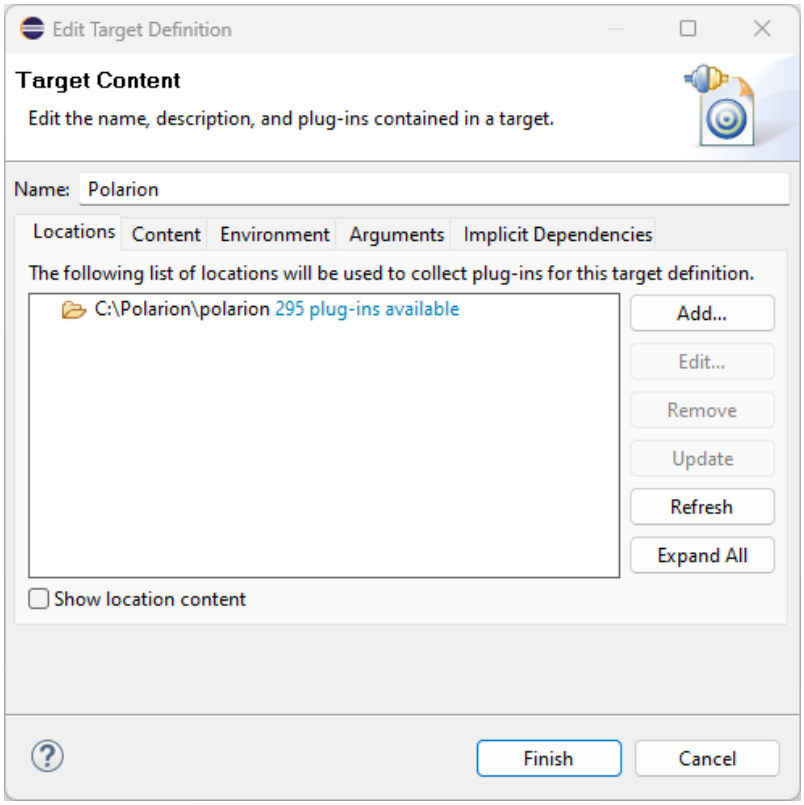


Figure WP-3: Confirm the Selected Path

10. The selected path and the number of discovered plug-ins available appear. Confirm that the path is correct and click **Finish**.{WP_4.png:|description=**Figure WP-4:** Select the Target Platform}
11. 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**. Th e 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.

NOTE: Servlets loaded by Polarion are cached in: `[Polarion_Home]\data\workspace\.config`. If this folder is not deleted before deploying a servlet extension (plugin) and restarting Polarion, then either the servlets will not be properly loaded, or the old ones will be loaded.

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:

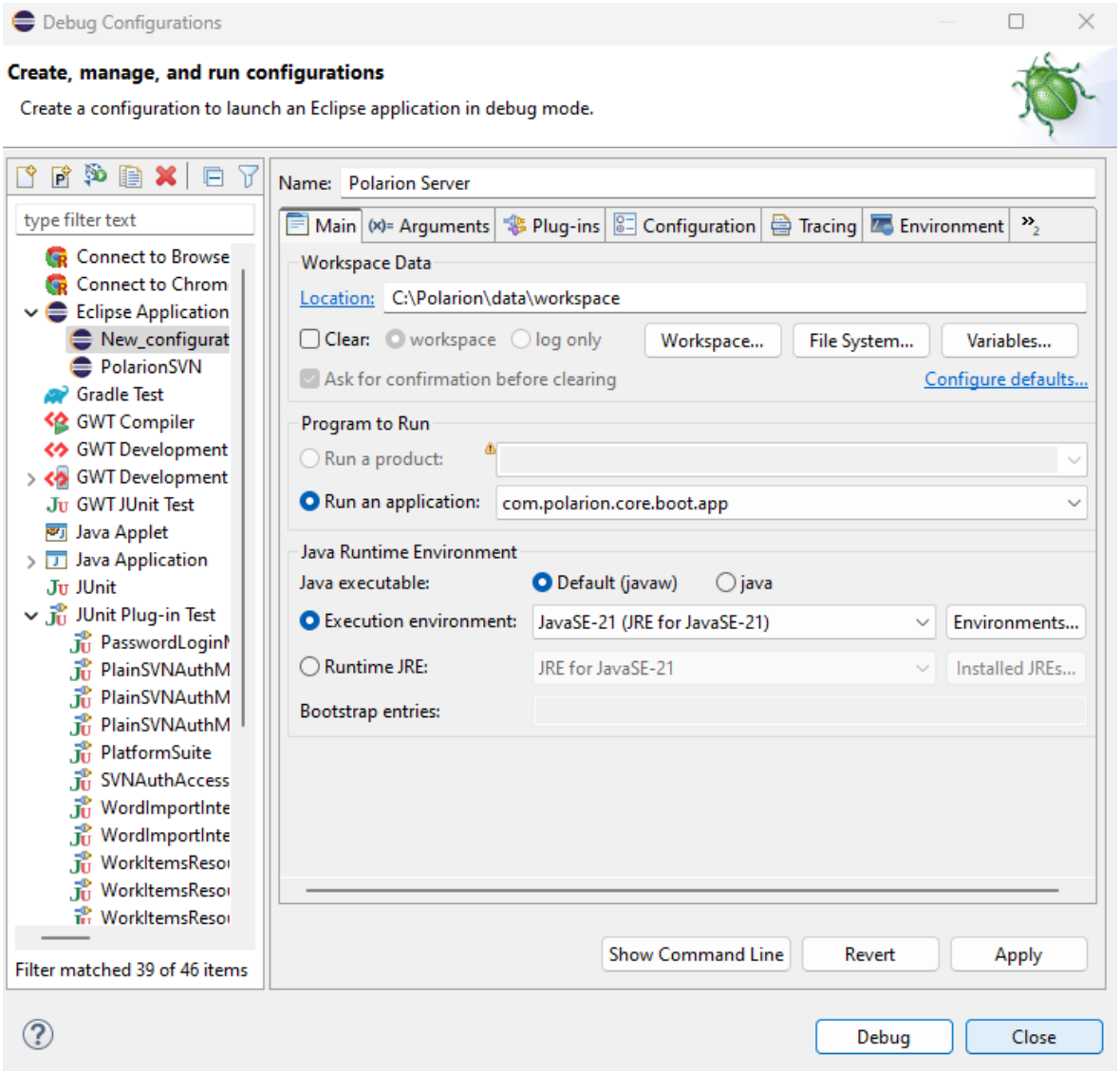


Figure EXEC-1: Debug - Main page

- **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.

4. Finally set your Runtime JRE. On the second, "**Arguments**" tab, set the following arguments:

In the **Program Arguments** section:

Windows:

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

Linux:

```
-os linux -ws gtk -arch x86_64 -appId polarion.server
```

In the **VM Arguments** section:

Windows:

```
-Xms1g -Xmx1g
-Dcom.polarion.home=C:\Polarion\polarion
--add-opens=java.base/java.lang=ALL-UNNAMED
--add-opens=java.base/java.lang.reflect=ALL-UNNAMED
--add-opens=java.base/java.net=ALL-UNNAMED
--add-opens=java.base/java.nio=ALL-UNNAMED
--add-opens=java.base/java.util=ALL-UNNAMED
--add-opens=java.base/sun.nio.fs=ALL-UNNAMED
--add-opens=java.base/sun.security.ssl=ALL-UNNAMED
--add-opens=java.management/java.lang.management=ALL-UNNAMED
--add-opens=java.xml/com.sun.org.apache.xerces.internal.dom=ALL-UNNAMED
--add-opens=java.xml/com.sun.org.apache.xerces.internal.jaxp=ALL-UNNAMED
```

Linux

```
-Xms1g -Xmx1g
-Dcom.polarion.home=/opt/polarion/polarion -Dcom.polarion.propertyFile=/opt/polarion/etc/polarion.properties
```

5. You must now change the parameters to the Polarion server based on your installation. You can check the settings with the following screenshot:

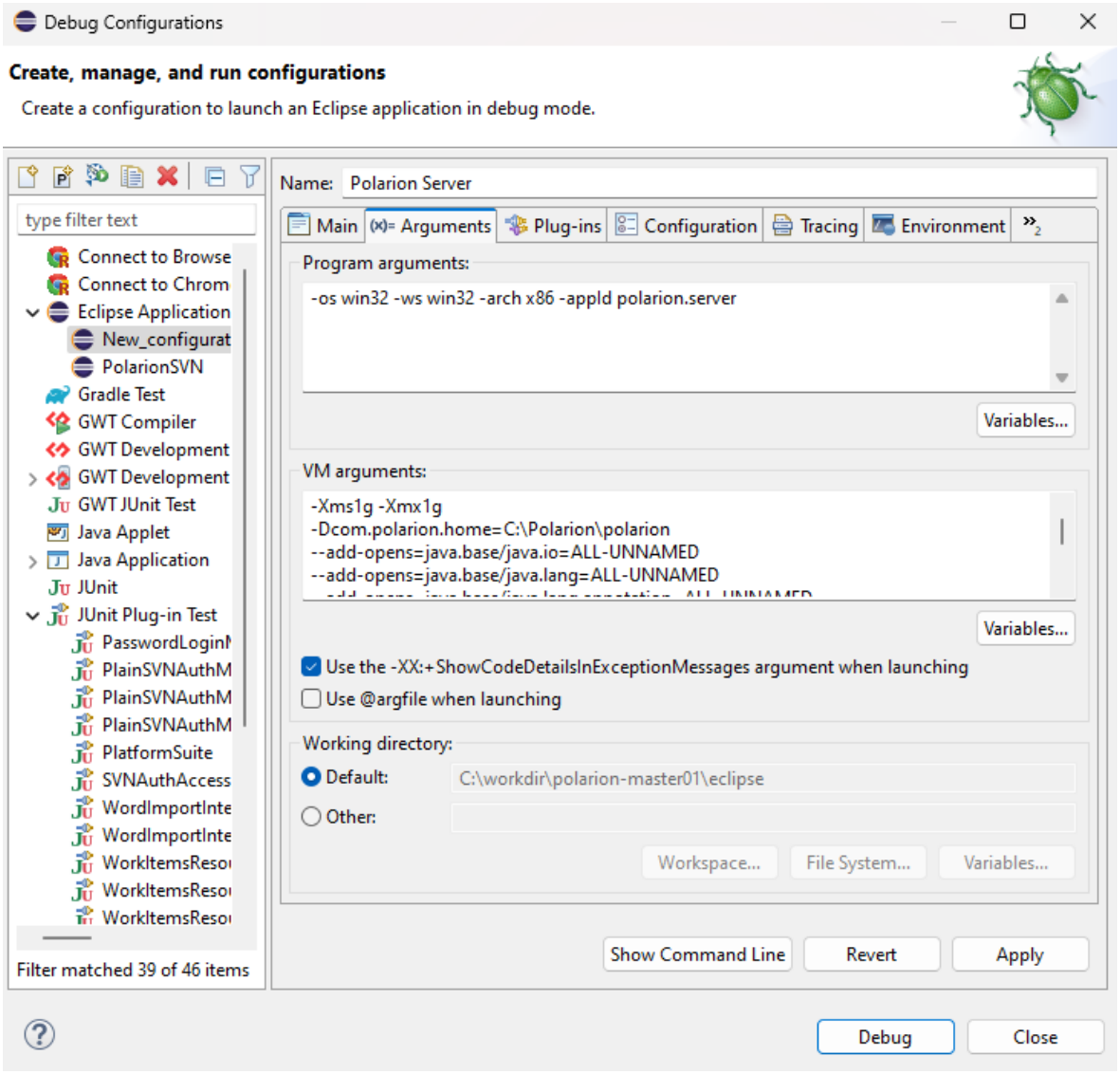


Figure EXEC-2: Debug - Arguments page

6. On the third "**Plug-ins**" tab, make sure, you have also selected "**Target Platform**" plugins.

7. Select all, and then click the **Validate Plug-ins** button. If there are some problems, uncheck the plugins which are in conflict.

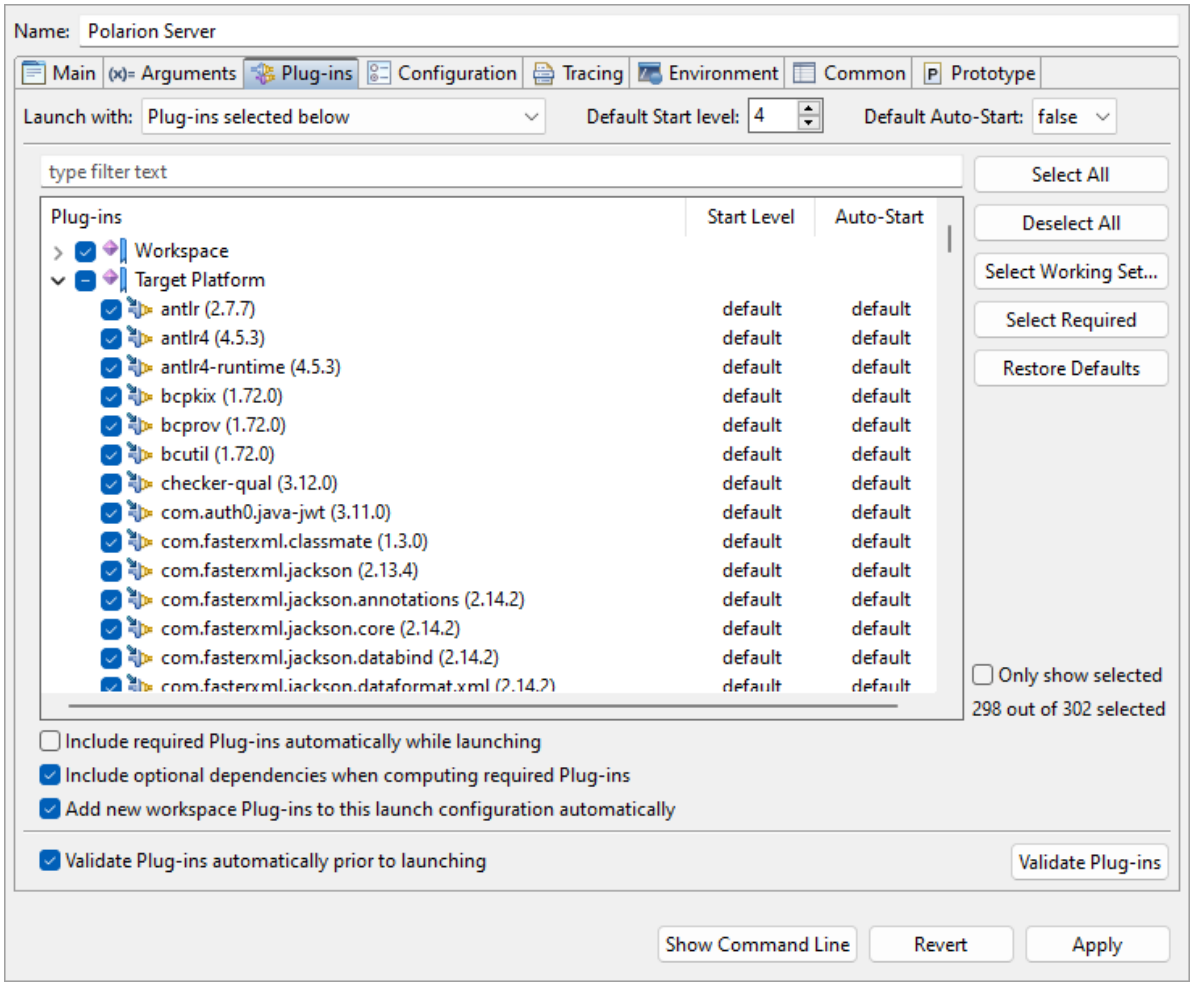


Figure EXEC-3: Debug - Plug-ins page

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