Table of contents

[START ECLIPSE 2](#_Toc535314428)

[GET YOUR GIT CLONE 4](#_Toc535314429)

[INSPECT GIT REPOSITORIES RESULT 7](#_Toc535314430)

[GET THE AVIX PLUGINS INTO YOUR PDE 9](#_Toc535314431)

[IMPORT PROJECT TEAM SET 11](#_Toc535314432)

[SETUP THE TARGET PLATFORM 16](#_Toc535314433)

[SETTING UP A DEBUG CONFIGURATION 18](#_Toc535314434)

[Importing a launch/debug configuration 18](#_Toc535314435)

In this document, we will outline how to get a suitable environment for AviX API development

up and running. Prerequisite: Having installed Eclipse (according to the “Eclipse setup”

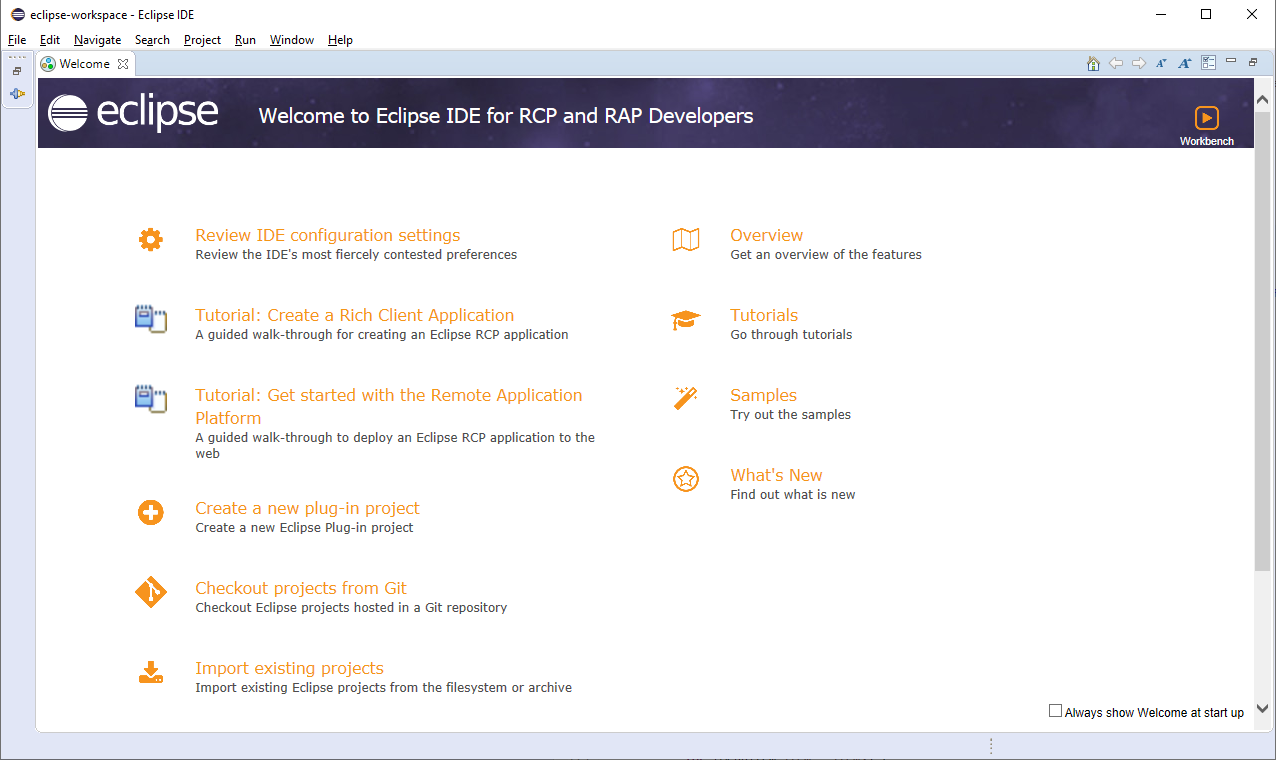
guide).

# START ECLIPSE

When you start Eclipse on a “fresh workspace”, which you are likely to be in if you came

straight from the “Eclipse setup” guide, you will be met with the so-called welcome screen

looking similar to this:

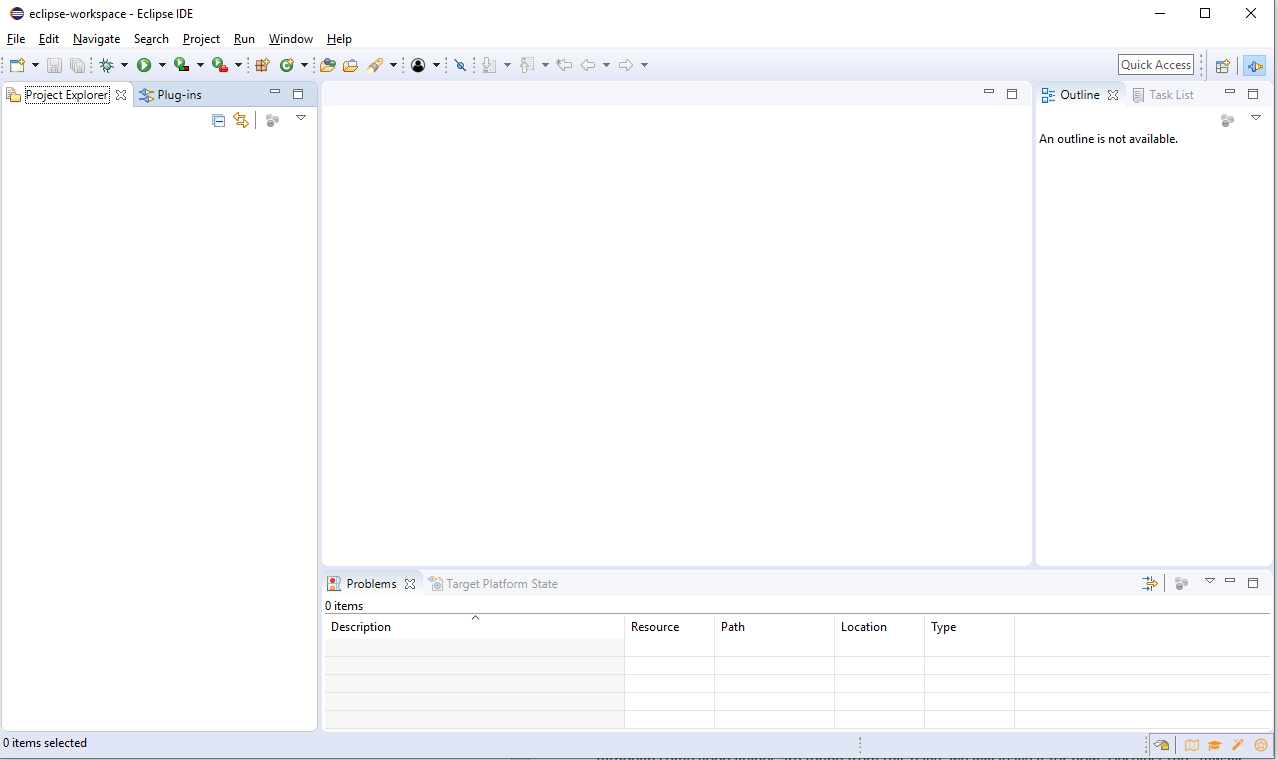


Although some good guides are found from this page, we will leave it for now. Deselect the “Always show Welcome at start up” at the bottom right, and then go to the *Workbench* by selecting the

“Workbench” icon in the top right corner. Now, Eclipse will probably bring you to the

*Perspective* called Plug-in Development (PDE). It will be empty, and looks something like

this:



3

A few fundamental concepts of Eclipse itself is present here right from the get-go. You need

to (get to) know the following terms and stuff when working with Eclipse:

- **Views**: these are the basic UI building blocks of an Eclipse application (like AviX; but

also Eclipse itself). In the rather empty image above, you have the “Project Explorer”,

“Problems”, “Outline” etc. All those are views.

- **Perspectives**: In the top-right corner, the icons representing the recently used

perspectives are present. In my case here, only the PDE perspective is visible and

active.

- **Editors**: The editor window is the big and totally empty window here. This is the place

where stuff is opened, for instance when selecting in a view. More about this later.

# GET YOUR GIT CLONE

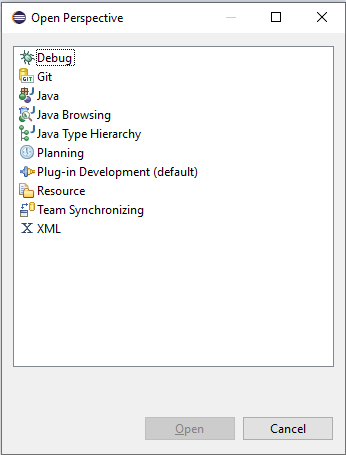
First, let’s get the AviX source code you need.

Since there at the time of writing is no real good API documentation of

AviX, the source code will be your best friend. If you are lucky, java components you need to study come with good java docs. If not, you’ll at least be able to look at the source, maybe copy a line or two, and of course also to debug it, to see what’s going on.

OK, open the GIT perspective: Click the little “Open Perspective”  button located in the top

right corner. You’ll be presented a dialog like this:



Select “Git” and click ok. You’ll now end up in the GIT perspective, looking like this. Also note

now in that the GIT perspective icon is there in the top right corner, for easy access later on.

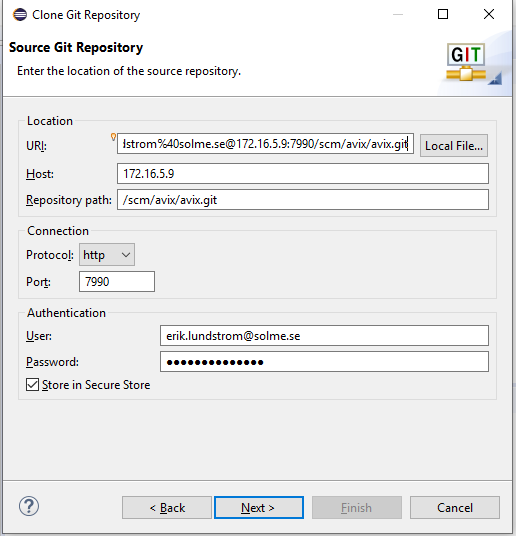
5

Now is the time to “Clone a Git repository”. Click that link in the “Git Repositories” view. Then,

click “Clone URI”.

On the next page, enter your credentials like I did here. To “Store in Secure Store” is a good

thing, so GIT won’t be nagging you for password all the time.



Copy URL to use (of course, you need to substitute your personal user name details there): http://firstname.lastname%40solme.se@172.16.5.9:7990/scm/avix/avix.git

6

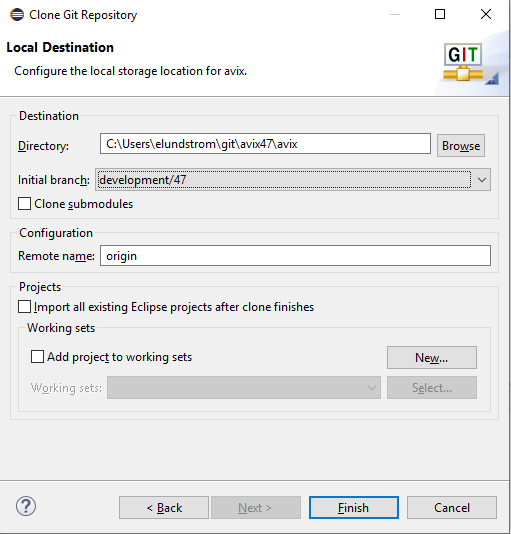
On the next page, you’ll select which *branches* to go for. Accept defaults (all) and go for

Next.

On the “Local Destination” page, you will select where on your computer/drive you will store the GIT

clone. This may occupy quite a bit of space, just so that you know. I did my selection like this,

but it is up to you where you want to store it.



7

Note that I selected “*development/47*” as my initial branch. At the time of writing, this is the

most current branch, so select it. (One can easily select other branches later on, so it is not a

big deal. That is one of the fundamental advantages with GIT btw – easy branch creation and

navigation).

Do **not** import existing projects. We’ll do it another way, just shortly. Click “Finish”, and eGIT

(Eclipse GIT integration) will start the process. This may take a very long time – hour(s) if on

a bad connection. So be patient!

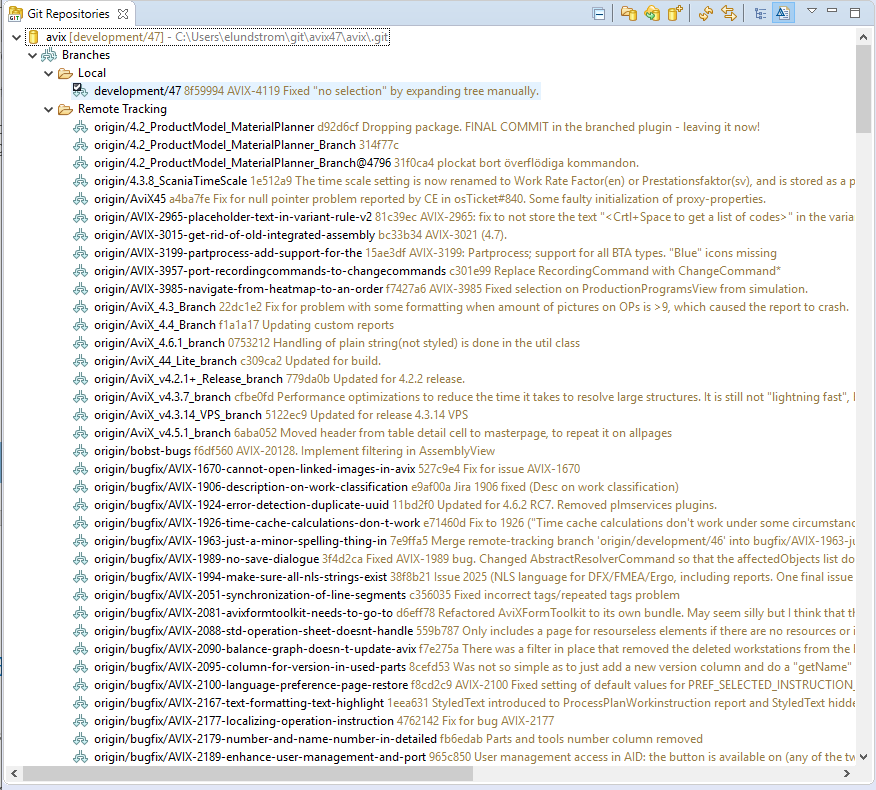
(Note: the reason why I also named the directory “…\avix47\avix” has to do with that I intend to have another GIT-clone when I am doing development for other branches, e.g. “master”. It helps to have dedicated clones for different development tasks (you can then launch the same Eclipse several times, and have one workspace dedicated to 4.7-development, and another for master-development.)

8

## INSPECT GIT REPOSITORIES RESULT

Hopefully, your GIT clone was created effortlessly, and you even had some time for a coffee or something.

Your Git Repositories view should now contain something like this:



You see the name of the Git repository (“avix”) here, as well as the Local branch you are

currently in (“development/47”). We selected that in the “clone git repository”, remember?

You also see all the existing “Remote” branches. Although we will not go into details

regarding Git here (there are excellent resources on the web for that), a better UI

organization IMHO is to select the little “arrow menu”, go for “Hierarchical Branch Layout”

and see the difference:

9



As you see, eGIT displays the branches as good as it can. All the branches directly under

“origin” is just because we didn’t position them in a good way from the beginning.

10

## GET THE AVIX PLUGINS INTO YOUR PDE

OK, so now you have all the source code cloned onto your hard drive. But if you jump pack

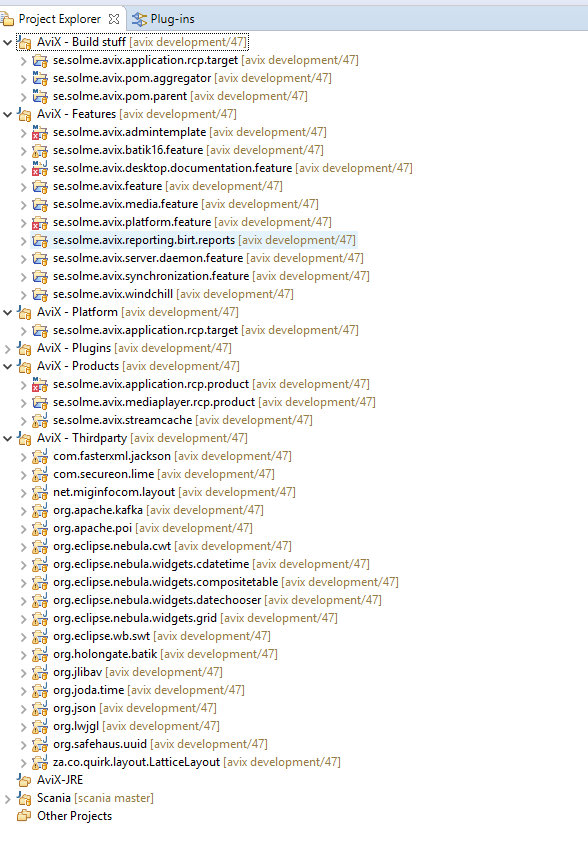
to the PDE perspective, the “Project Explorer” will still be empty. That will not remain this way

for a long time – this is where we want all the AviX plugin projects to be presented.

In order to get to that, you will need to “import projects” into your so-called working directory. That can be done directly from the eGIT side (by selecting projects in the Git Repositories and importing them), but we will not do it like that.

Instead, you should ask a nice colleague to perform a “Team Project Set” export from his/her Eclipse workspace, which you can import. What that basically means is that you will have exactly the same projects present in your Project Explorer in the end, as that your colleague has.

Take my workspace for instance. My Project Explorer looks like this:



I will briefly explain a bit on this setup. The top nodes here are called “Working sets”, and it is just a convenient way of organizing your projects (which in turn can be of different natures).

The ones visible here are the ones that you will not have to care that much about really. Just some words on them.

"***Build***": stuff needed for our build server. (As a new developer, you will not normally need to care about these.)

"***Features***": plugins are included in different *features* in Eclipse. We have a couple of features in AviX, and there will probably be more in the future (we need to layer the application even further, especially when we have several products (normal AviX, web AviX, etc) that partially use the same components)

"***Platform***": target platform stuff. More on that in a while.

"***Plugins***": This is really the core, since this is where all normal java projects go. As you see in the image, it remains collapsed, since there are so many projects in this one.

"***Products***": maps somewhat to the concept of a runnable application/product. You need not to care about this as a new developer.

"***Thirdparty***": the 3rd party plugins that we use for AviX. You need these to run AviX. Contains different components not developed by us.

"***JRE***": empty in my case, so not needed really.

"***Scania***": Simply there in my config since I seem to have done some customization work lately for that customer.

So what does this all mean? Well, all you see above are projects of different natures (java, feature projects etc) that I have checked out from the GIT repositories. You can and should do the same! Ask your colleague to give you that “Team Project set export”, and then proceed to the next step.

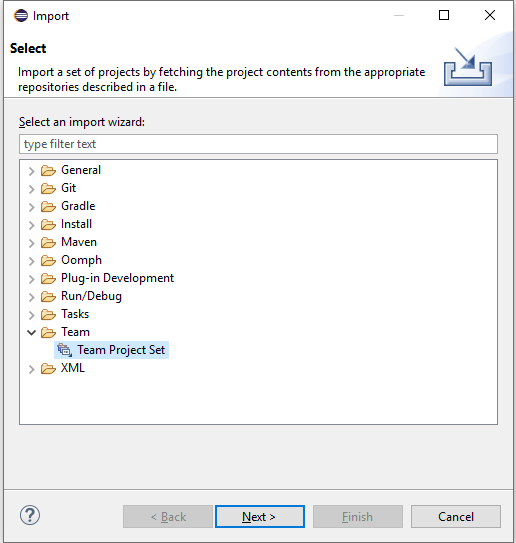
## IMPORT PROJECT TEAM SET

Prerequisite: that you have received a project set export file from someone. It is likely to be called “projectSet.psf”.

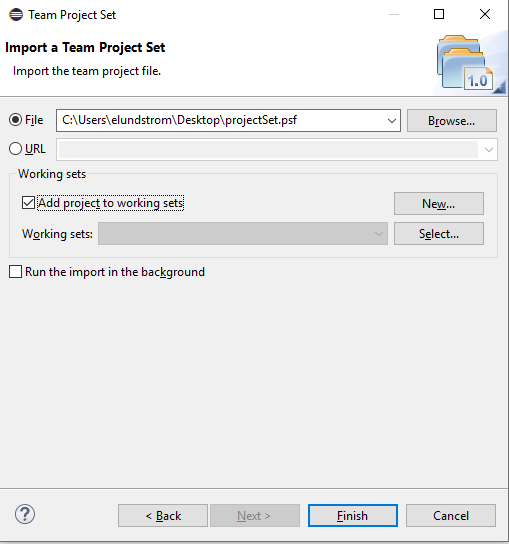
First of all, you need to substitute the credentials of the user who exported that file, to use your own credentials. Notepad++ or any competent editor will quickly help you to do the search and replace. The file looks like this from the beginning (all the “erik.lundstrom” stuff is what needs replacing):



Having done that, go to the PDE perspective in Eclipse and launch “File->Import…” in Eclipse, and select this import:

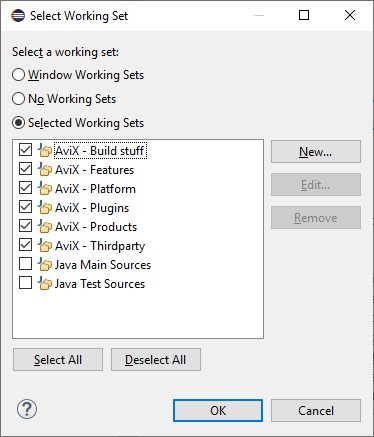


On the next page, do like this:

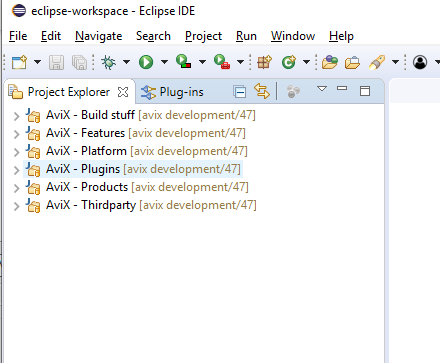


Click Finish. This will take some time. During the progress, you should see lots of projects appear in the Project Explorer.

Once done, you will probably have an overwhelmingly big flat list of projects. So, it does not look just yet as it did in your colleague’s workspace. To change the presentation to use working sets, do like this: Using the little “arrow menu” in Project Explorer, select “Select Working Set…”. You will be presented with this. Make the selections as in the image, and press OK.



Now, using the little arrow menu again, change “Top Level Elements” to Working Sets. That should result in the following:



Ha11

12

13

14

15

And that’s it – this completes the GIT interaction and project setup process for now.

# SETUP THE TARGET PLATFORM

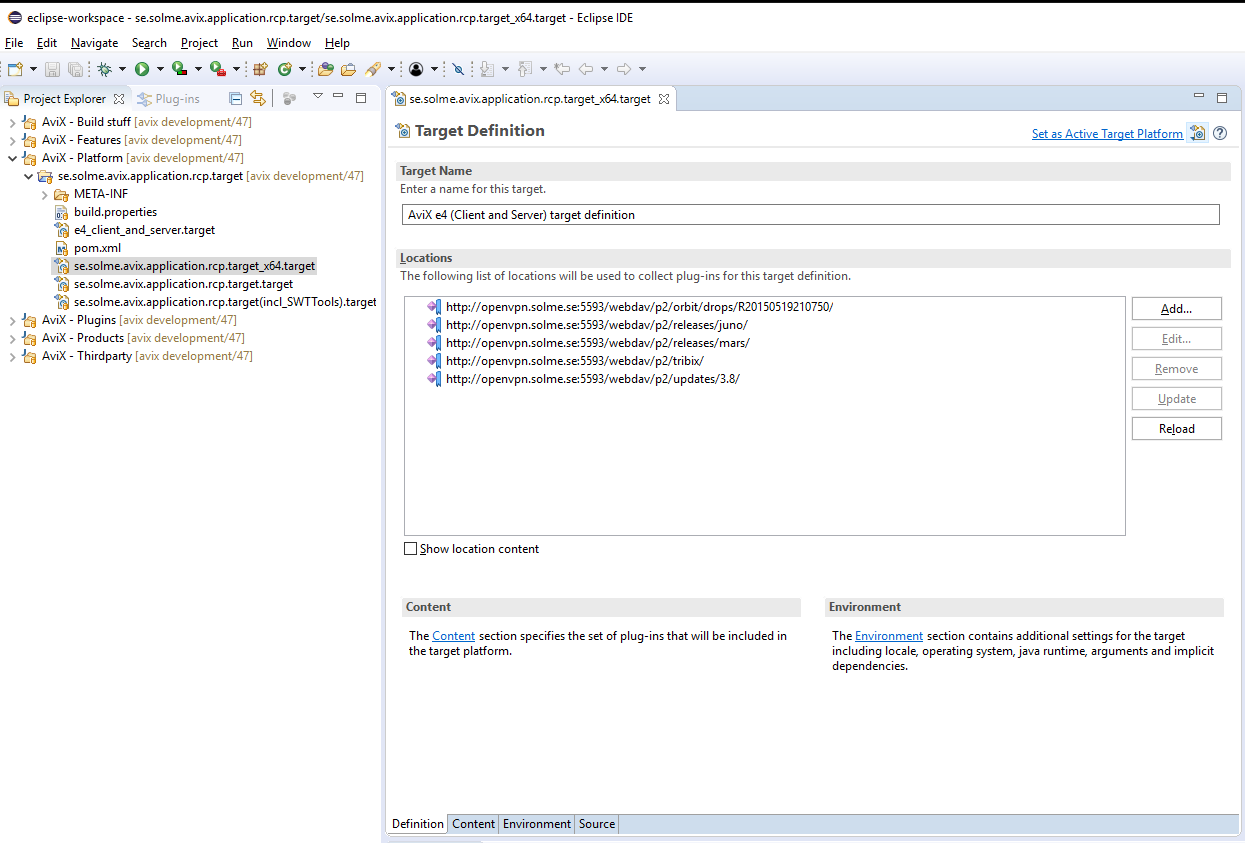
Having completed the above steps, your Workspace is configured with the GIT connection and also has all the required projects imported into the workspace.

If you have a look in the “Problems” view however, you’ll notice that not many things are functional yet. This is because the numerous (java) projects don’t compile successfully.

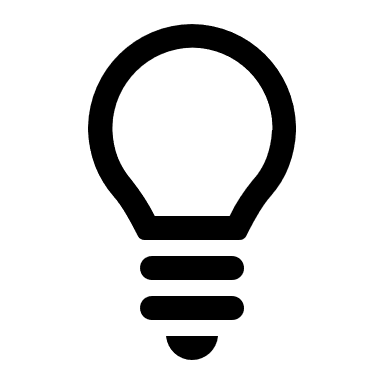
We will resolve this by setting up your so-called “*target platform*”. To put simply, it is the

code base with all the plug-ins that our precious AviX plugins needs to run. The target platform will load all the pre-requisite features and plugins, and actually also the desired java. The target platform is the fundament which all of the rest relies upon, and compiles upon!

Fortunately, there is a so-called “target definition” file already existing, which can conveniently be used to set up the target platform of your Eclipse workspace. Locate the following file, and double-click it to open:



Please note that this make take a while - see the progress indicator in the bottom bar of Eclipse. What it will do is to load the definitions from our server (the openvpn.solme.se stuff).

You are likely to need credentials to access the “openvpn.solme.se” server. You should have received these from someone knowledgeable at Solme!

Once loaded, you may have a look on the different tabs in there, but please do not change anything. It should be this way!

To set the target platform, you need to press "Set as Active Target Platform", in the top right corner of the editor. Now...you can probably go and tale a cup of tea or something, because this will take time. Eclipse will download everything needed from our servers and store them "somewhere" on your computer. You don't really need to know where - it's internal Eclipse stuff.

Once it has been done, the expected result is that everything compiles, meaning that problems in the Problems view should be zero (or maybe a few). Also, if eye-balling projects in the Project Explorer, they should not have the red error decorations.

# SETTING UP A DEBUG CONFIGURATION

Now that we’ve come this far, it would be great to be able to debug AviX, wouldn’t it? Let’s do

it!

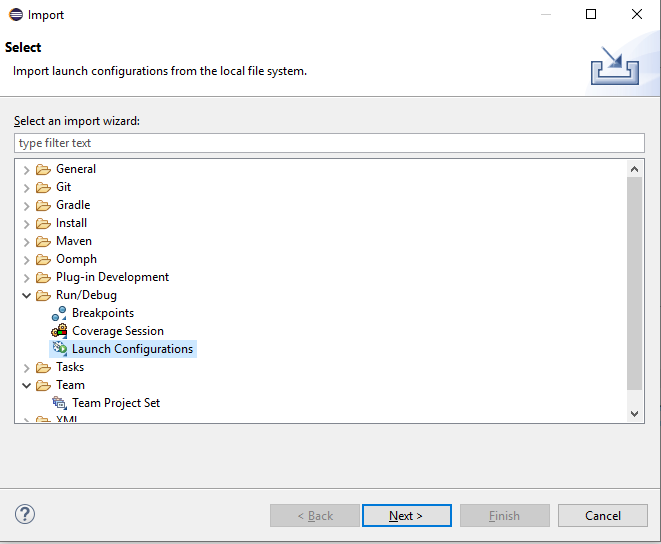
Once again, we will make use of a configuration that someone else exported for your convenience. But just some brief words on debug configurations first.

* Once can easily create configs from scratch, pressing the little arrow menu adjacent to the debug icon  in the toolbar. This is also the place you go to edit existing profiles, once they exist (right now you probably have none)
* A debug profile is simply a set of plugins you would like to debug/run at the moment. All the external plugin dependencies will be “pulled” from the target platform that we setup in the former section.
* One can have many debug profiles existing at the same time. Good, for example when one wants to have:
  + One “clean” config which is “plain standard AviX”
  + One config which is “AviX plus customization 1” on top of that
  + Another config which is “AviX plus customization 2” on top of that
  + etc…

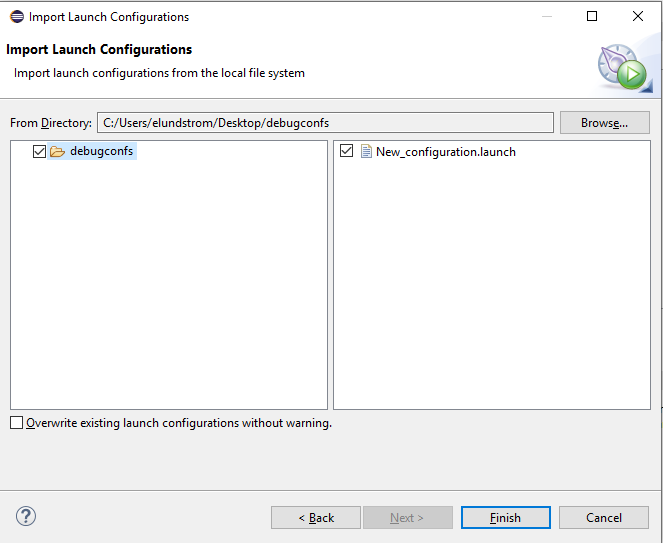
## Importing a launch/debug configuration

Prerequisite: that you have received a launch export file from someone. It is called “<conf\_name>.launch”.

Use “File->Import…” in your Eclipse workspace. Select the following:

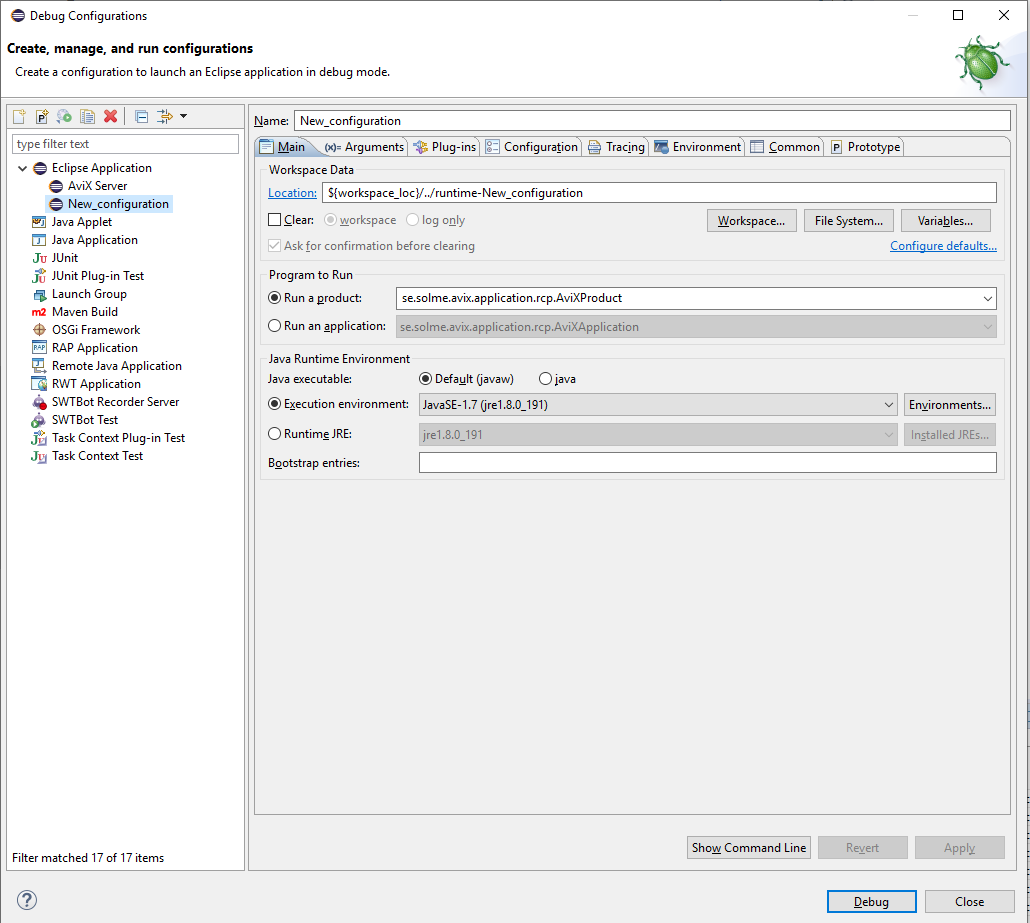


On the next page, it anticipates a directory as you see. (This is since it wants to facilite the import of multiple configurations at the same time, I guess). So put the “.launch” file in a directory of its own, and select that.



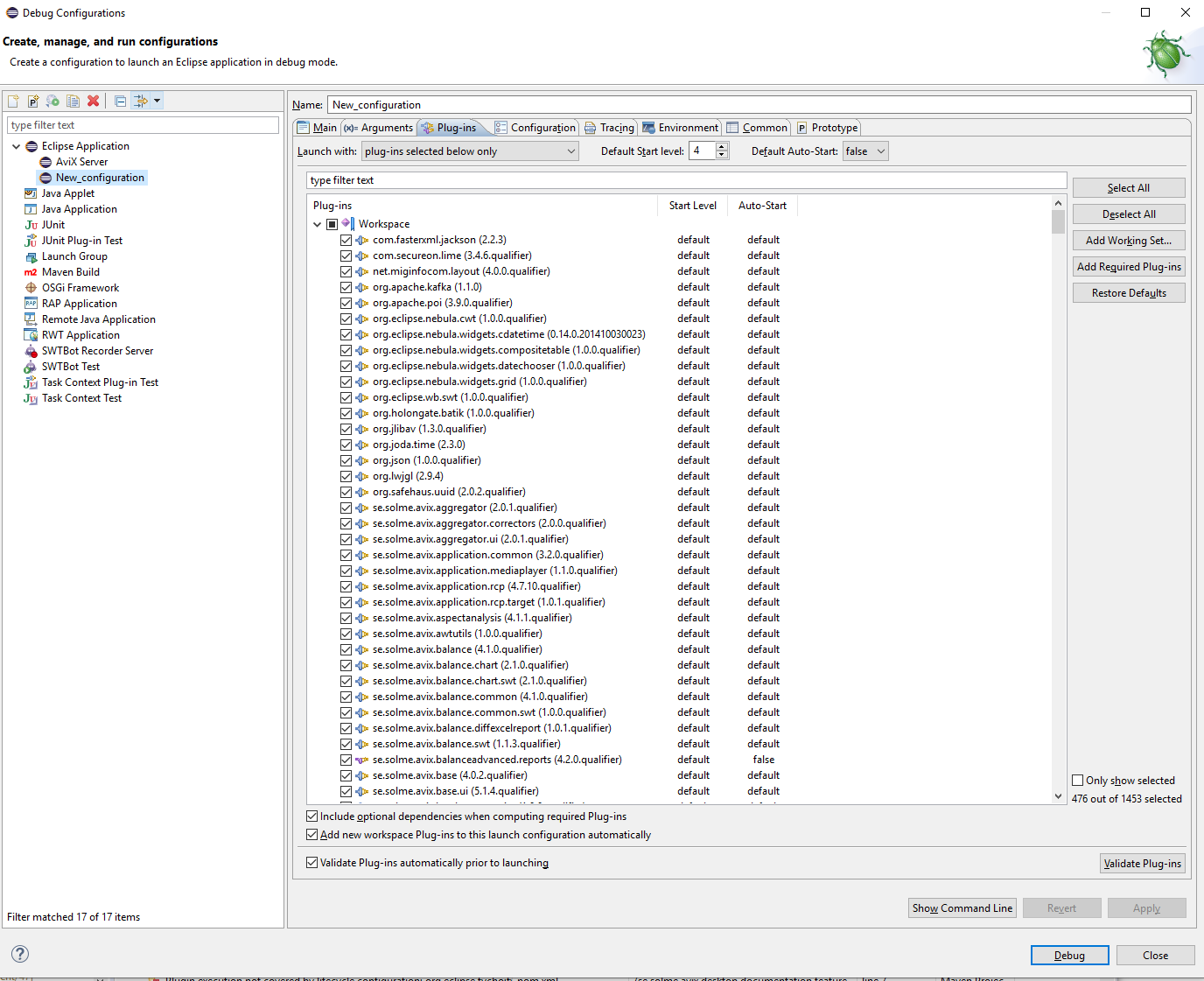
Select Finish when having made the selections above.

Back in the Eclipse workspace, select the Debug arrow menu, and go for “Debug configurations”. Under “Eclipse Application”, the imported one should appear:



You could flip through the tabs there, but I will not explain them at length. Note however that a good feature is that you can copy/clone an existing configuration later on, when you are to create another one. Always good to start out with something that works!

Regarding the “Plug-ins” page: this is the most important one. I’d suggest you have a look there:



As you see, most “Workspace” plug-ins are selected in this configuration by now (some of them were maybe omitted in the export you received). Also, note that “Target Platform” ones have already been added.

What you should do now is to validate that the total set up of plugins and their dependencies are ok. Press the “Validate Plug-ins” button. The expected result is that no problems were found.

Actually, now you are ready to launch AviX for the first time in debug mode! Press the “Debug” button at the bottom of this dialog.

Expected result: AviX will launch in debug mode.

* If this is your first ever run of AviX, you will be prompted to enter licensing details, and activate your license. That info should have been submitted to you already – if not, ask someone.
* After having launched a debug config once, it will be available directly by pressing the Debug icon directly (no need to go through the dialog again).