A Practical Agent Programming Language

Installation Guide

Contents

1 Introduction

This document contains a concise description of how to run and develop applications for 2APL. Please see the User Guide for a more detailed introduction to the 2APL platform as a whole. In what follows we assume that the reader uses Eclipse as the working environment.

2 Running the 2APL platform

The 2APL platform can be run in two ways: directly (from the command line) or using Eclipse. When using Eclipse, the user will receive useful debugging information.

2.1 Running 2APL directly

The 2APL platform can be started in:

Windows by double clicking the file `2apl.jar`, or alternatively, typing `java -jar 2apl.jar` in a command prompt window at the `2APL` directory

Mac OS by double clicking the file `2apl.jar`

Linux/Unix by typing `java -jar 2apl.jar` to a prompt in the `2APL` directory

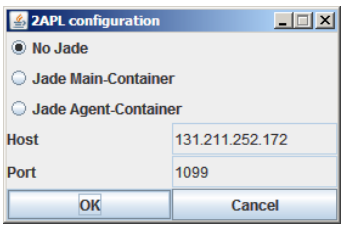
On every operating system, the 2APL platform can be invoked from the command prompt. For help the following command can run `java -jar 2apl.jar -help`. Here is the output:

```
$ java -jar 2apl.jar -help

2APL (A Practical Agent Programming Language) Interpreter

Usage: java -jar 2apl.jar [-nogui] [-nojade] [-help] [<path to MAS file>]

Options:
   -nogui   do not open graphical user interface; start the MAS immediately
   -nojade  skip JADE configuration and run in standalone mode
   -help    print this message
```



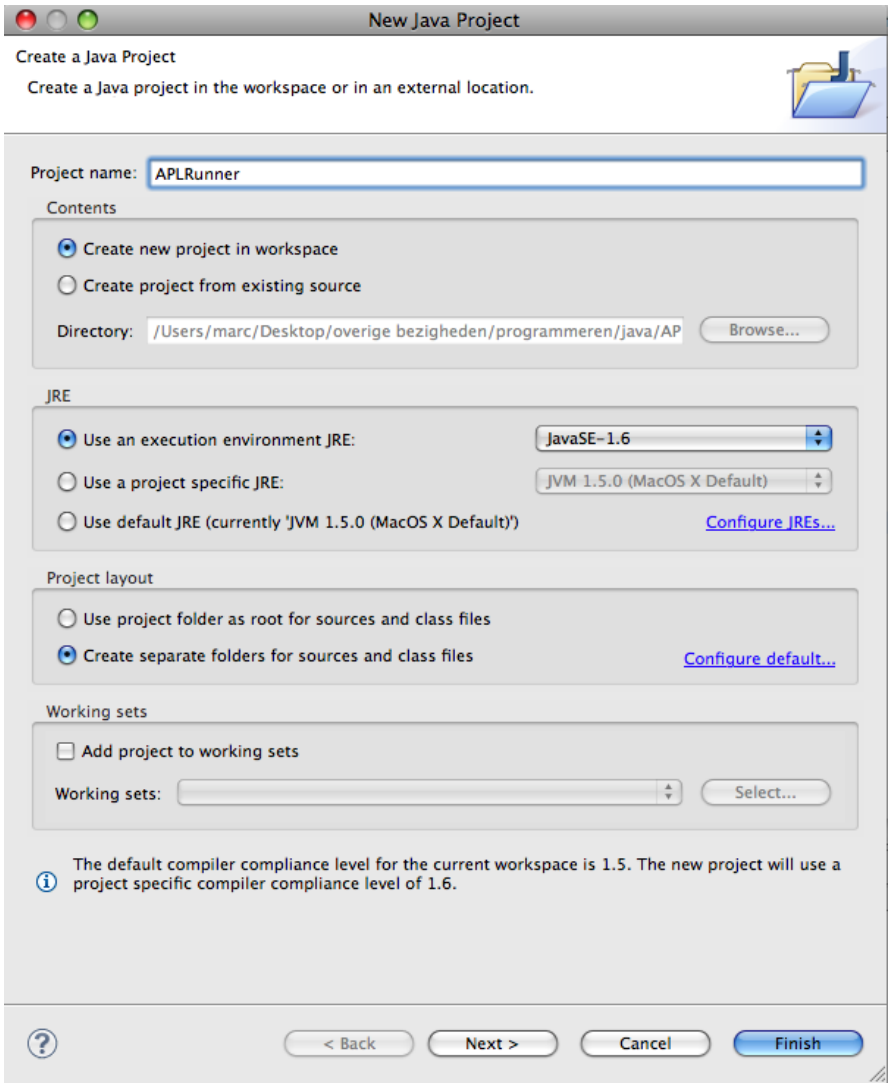
As illustrated in the image above, one needs to select a configuration option when starting 2APL without arguments. The first option allows the use of 2APL platform in a stand-alone mode and the last two options

start 2APL on top of the Jade [**?**] platform. The last two options are useful when one wants to run a multi-agent program (in a distributed manner) on different machines. In this user guide, we focus on the stand-alone run mode of the 2APL platform.

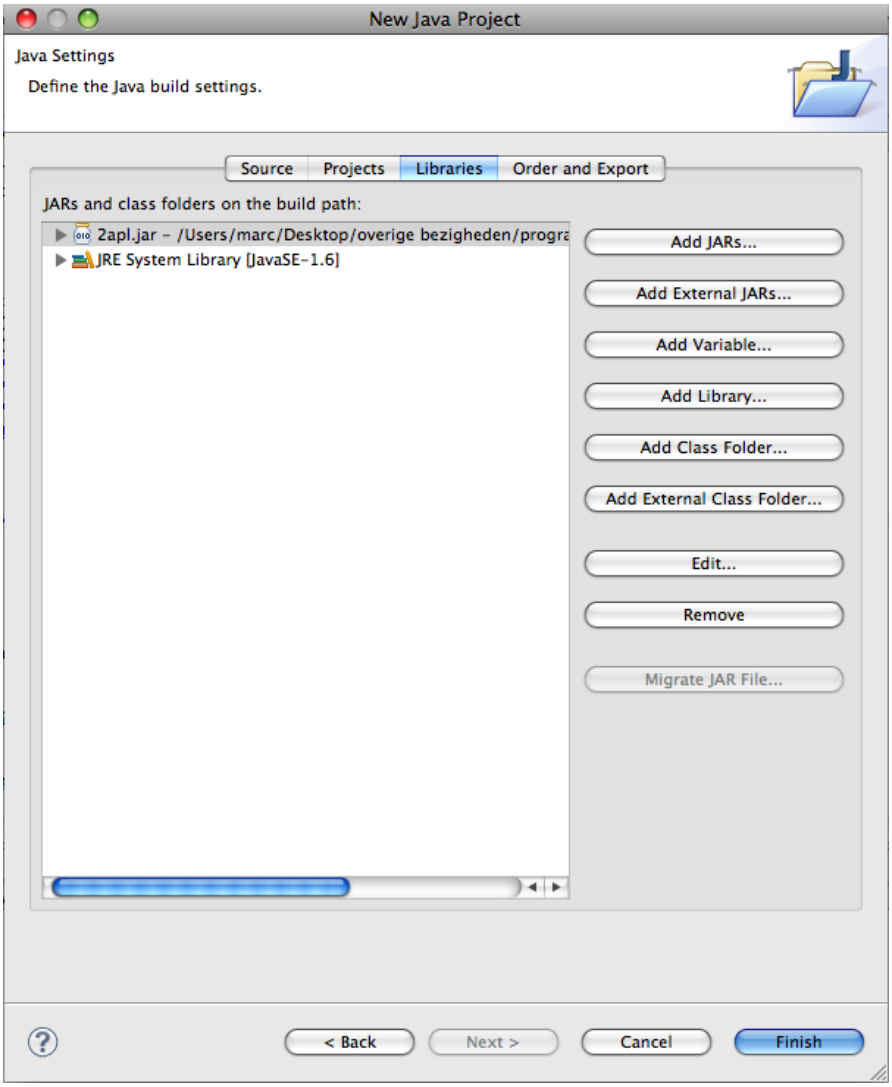
Alternatively, one may load and start the multi-agent system directly from the command line. One can also start the interpreter without a graphic user interface. To see the help on the available options execute `java -jar 2apl.jar -help` in the command line prompt.

2.2 Running 2APL using Eclipse

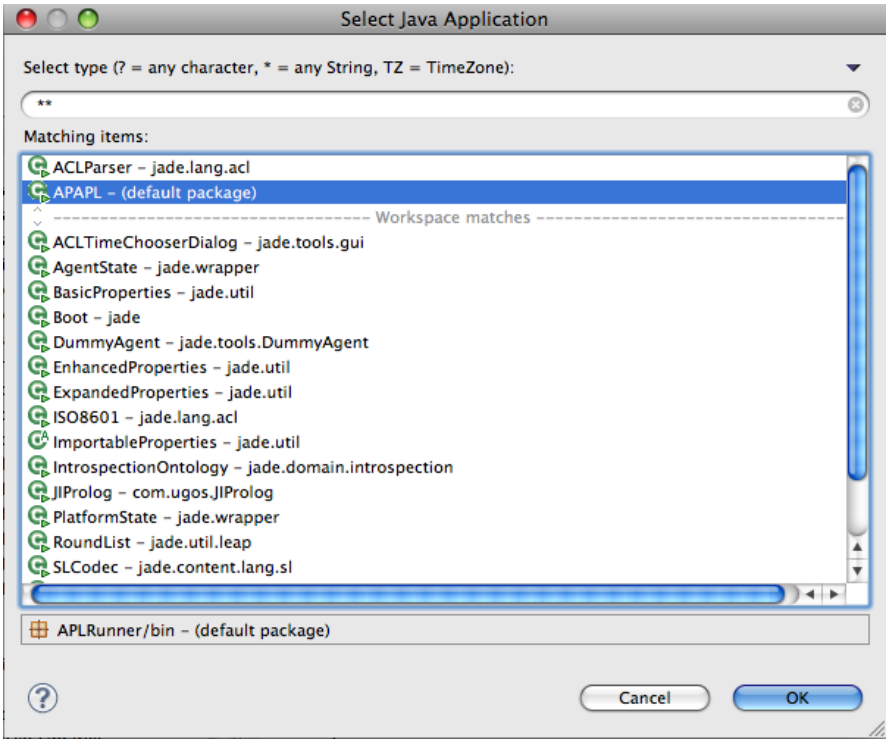
Create a new java project in Eclipse: `File > New > Java Project`. Choose a project name (we choose "APLRunner") and click `Next`.



On the next screen, go to the tab `Libraries` and click `Add External JARS...` Select `2apl.jar`, the JAR that is included in the 2APL platform. Make sure that the `lib` folder is in the same directory as this file as well, because 2APL needs these to run correctly. Add the JAR and click `Finish`.



In the Package Explorer, right-click on the project you have just created and go to `Run as > Java Application`. In the dialog that opens you are asked to select the main class. This will be the class that we run, which is the main class called `APAPL`.



Now the 2APL platform will launch, and debugging information will be printed to the screen. For example, if you run the `ircbot` example it will output all the data that is being received from the chatserver.

3 Developing for the 2APL platform

3.1 Required components

To develop an application for the 2APL platform, you need to create at least three files. Please see the `examples` directory for examples of these files.

- The MAS specification file. This is a text file with the extension `.mas` that contains an XML specification of the multiagent system. The name and path to the source of all agents and environments that start in the application are denoted here.
- A 2APL agent file. This is a text file with the extension `.2apl` and contains the 2APL code of the agent.
- An environment file. This is a runnable JAR (with the extension `.jar`).

Since the environment file is a compiled Java program, the developer needs to build this separately. We will now see how an environment can be built using Eclipse.

3.2 Building an environment using Eclipse

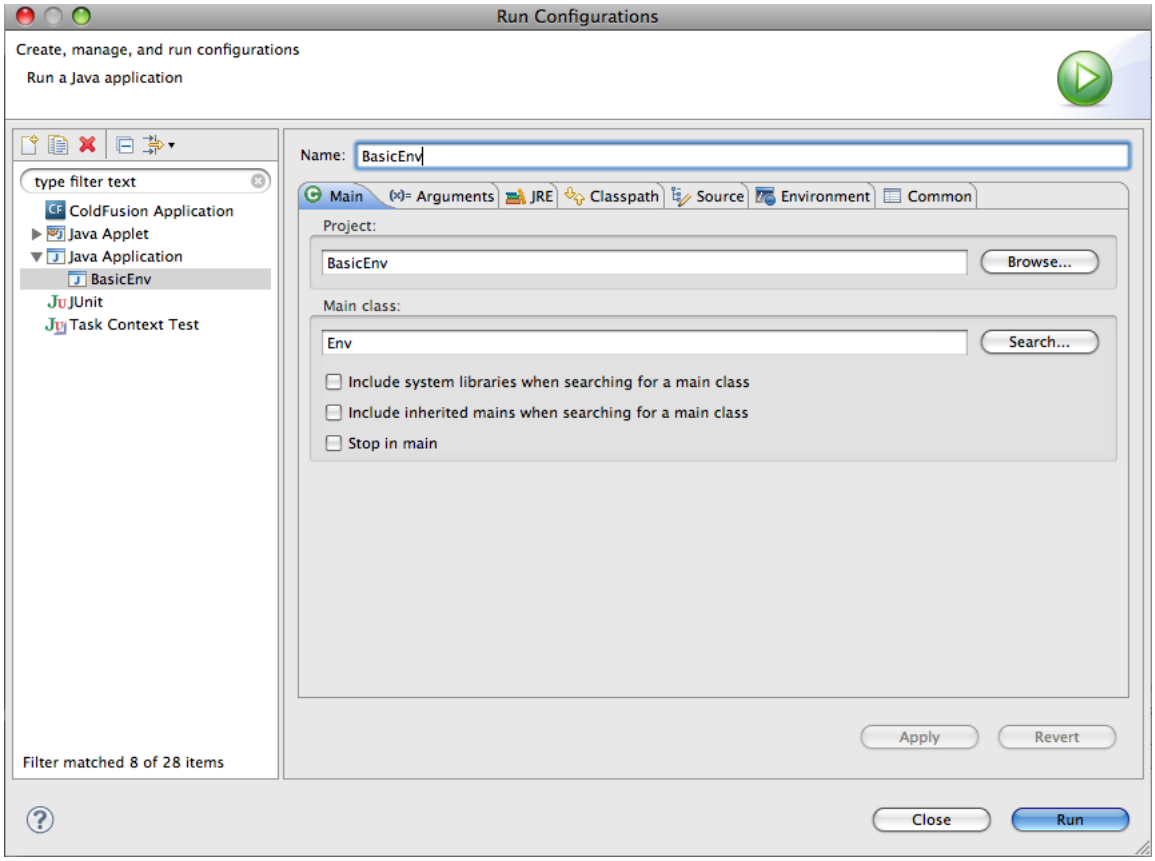
As an example, we will compile the environment that is contained in the example `basic 2apl setup`, but the reader is of course free to alter this example in any desired way. The environment is contained in the directory of this example and is called `Env.java`.

Create a new java project in Eclipse in the same way as explained in section 2.2. Make sure you don't forget to add `2apl.jar` as an external JAR. We call this project `BasicEnv` but you are free to choose the name you like.

Now we will add the environment file to the project. The easiest way to do is, is to simply copy `Env.java` into the source folder of the project: go to the workspace directory of Eclipse, then copy `Env.java` into the `src` folder of the project you just created. Now go back to Eclipse, right-click on the project and click `Refresh`. You should now be able to edit Env.java, but we will not do this now. Please see the comments in this file for more explanation on the working of the environment itself.

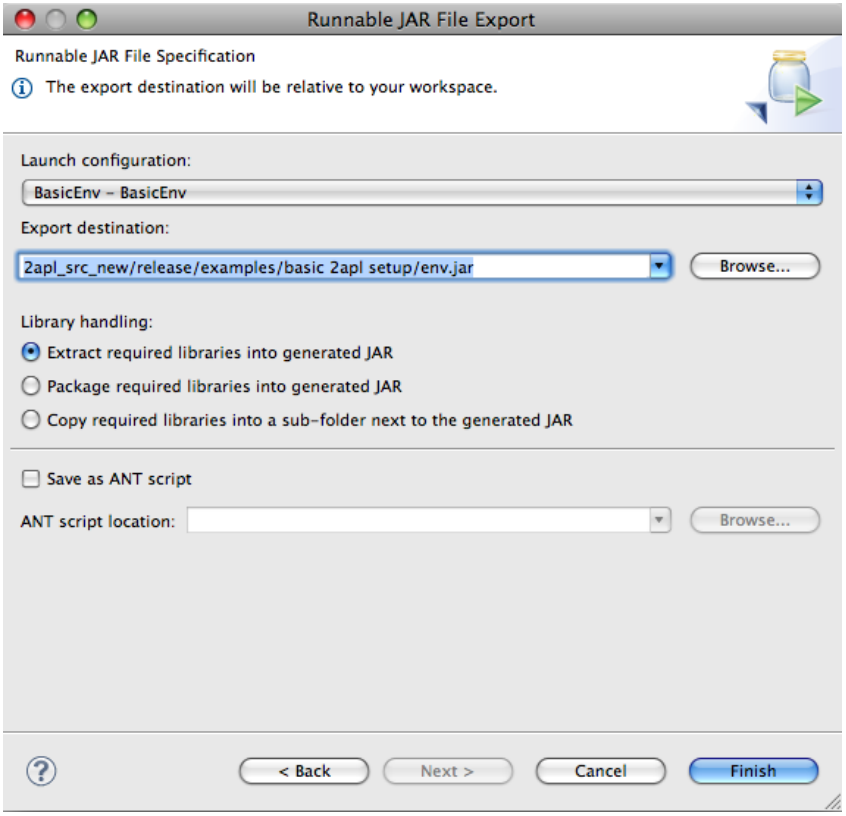
We will now create the runnable JAR. For this we need to create a run configuration. Go to `Run > Run Configurations...`, click on `Java Application` in the left panel and press the `New` button in the top left of this panel. Choose a name for your run configuration (for example your project name) and make sure that your project is selected as the `Project`. If not, click `Browse...` and select your project. Now make sure that the main class of your run configuration is the environment, in the case of our example this is `Env`. If this is not the case, select `Search...` and choose the class of your environment.

Click `Apply` and then `Close`.



The idea of this run configuration is that we tell Eclipse that we want the runnable JAR to run the Environment class when it is called. If we then use the environment in the 2APL platform, it will choose the correct class as the environment class.

We are now ready to actually build the JAR. Right-click on your project and click `Export...`. Select `Java > Runnable JAR file` in the dialog that opens and click `Next`. In the next screen, select the run configuration that you have created in the previous step as the launch configuration. Choose the export destination of the JAR. This should be the place where the rest of your files (the MAS specification file and the 2APL files) are also located. Make sure "Extract required libraries into generated JAR" is selected and click `Finish`.



You will see several warning which you can ignore. Congratulations, you have just built a 2APL environment! Now you can now simply run the MAS file as explained in Chapter 2.