Getting the Photran Development Sources from Git

and Creating Launch Configurations

Last modified July 22, 2013

This describes how to check out the source code for the current *development* version of Photran. At any given time, this is the version of Photran that will be released the following June.

The source code you check out using this document is *not* guaranteed to be stable. There are rare points in time when it may not even compile. So, if you have problems following the instructions in this document, please ask for help on the ptp-dev mailing list – https://dev.eclipse.org/mailman/listinfo/ptp-dev.

Before you begin...

1. Photran's source code *will not compile* if you are using the wrong version of Eclipse! You will need to install *Eclipse Classic*, also known as the *Eclipse SDK*, which includes the Eclipse Platform, Java development tools, and Plug-in Development Environment. (Since Photran is a set of Eclipse plug-ins, you need a distribution of Eclipse that includes the Plug-in Development Environment.) What version you need depends on the time of year.
   - Between January and June, you need to be running the most recent milestone build (in preparation for the upcoming release, which occurs in June). To download the latest milestone build, browse to
     http://download.eclipse.org/eclipse/downloads/
     and click on the link for "4.x Stream Stable Build." This will take you to a page where you can download the Eclipse SDK. (For example, in January–June, 2013, you will need the 4.3 Stable Stream Build for the upcoming Eclipse Kepler release train.)
   - Between July and December, you can use the latest release. Download Eclipse Classic from
     http://www.eclipse.org/downloads/
     (For example, in June–December, 2012, you will download Eclipse 4.2 Classic.)
2. You will need to be running a Java 6 or later Java Virtual Machine (JVM).
3. *(Optional)* You may want to install the following Eclipse plug-ins, although none of them are necessary:
   (a) **Subclipse** provides Subversion support – http://subclipse.tigris.org/
   (b) **FindBugs** is a static analysis/bug detection tool for Java – http://findbugs.sourceforge.net/
   (c) **EclEmma** is a tool which helps you determine the code coverage of your JUnit tests – http://www.eclemma.org/
   (d) **Metrics** provides code metrics for Java code – http://metrics.sourceforge.net/

Part I. Install the CDT SDK, EGit, and Photran

First, you will need to install the C/C++ Development Tools Software Development Kit (CDT SDK), EGit, and the most recent stable release of Photran itself. Photran reuses some parts of CDT, and the CDT SDK is required to view the source code for CDT. EGit provides support for the Git distributed version control system in Eclipse; it is required to access the Git repositories at the Eclipse Foundation which contain Photran and CDT's source code. Installing the

latest release of Photran ensures that Fortran files will be properly syntax highlighted (among other things), which is helpful when writing tests. It is also required when setting an API baseline in Part II below.

1. Start Eclipse. Inside Eclipse, click on the "Help" menu, and choose "Install New Software..."
2. In the "Work with:" combo box, choose the update site for the latest Eclipse release train. As of July, 2013, this should be labeled "Kepler – http://download.eclipse.org/releases/kepler"
3. Expand the "Collaboration" category
4. Under "Collaboration," check the box next to "Eclipse EGit"
5. Expand the "Programming Languages" category
6. Under "Programming Languages," check the box next to "C/C++ Development Tools SDK" (Be sure the name ends with "SDK"!)
7. Under "Programming Languages," check the box next to "Fortran Development Tools (Photran)"
8. Click "Next"
9. The wizard will show the complete list of plug-ins to be installed; click "Next"
10. Accept the terms of the license agreement, and click "Finish"
11. Installation may take several minutes. Restart Eclipse when prompted.

Part II. Set an API Baseline

Eclipse plug-ins are required to follow very strict rules when making changes to their application programming interface (API). The Eclipse Plug-in Development Environment's API Tooling component enforces these rules. You must configure it with an API baseline so it can compare the current source code with the latest stable release, in order to detect API changes.

12. In Eclipse, if you are running Windows or Linux, click on the "Window" menu; if you are running Mac OS X, click on the "Eclipse" menu. Choose "Preferences" from the menu.
13. Expand the "Plug-in Development" category, and choose "API Baselines"
14. Click "Add Baseline..."
15. Enter "Default" for the name
16. Click "Reset"
17. Click "Finish"
18. Click "OK"
19. If prompted to "Do a full build now?", click "Yes"

Part III. Clone the Photran Git repository and check out Photran's source code

Important: *If you already have an earlier version of the Photran source code (e.g., if you checked it out from CVS before we moved to Git), you must delete the existing projects from your workspace. The Git import wizard will not overwrite them; it will give an error message and fail.*

20. Switch to the "Git Repository Exploring" perspective. (From the "Window" menu, choose "Open Perspective", and "Other..."; choose "Git Repository Exploring", and click "OK.")

21. From the "File" menu, choose "Import..."

22. Under "Git", choose "Projects from Git", and click "Next"

23. Select "URI", and click "Next"

24. In the URI field, enter one of the following.

    - Most people will enter
      `git://git.eclipse.org/gitroot/ptp/org.eclipse.photran.git`
    - If you are a committer at the Eclipse Foundation, enter
      `ssh://username@git.eclipse.org:29418/ptp/org.eclipse.photran`
      replacing `username` with your committer username.

25. Click "Next"

26. All branches will be selected by default; click "Next"

27. Make sure the local directory is OK (or change it if you want your local Git repository stored somewhere else); then click "Next"

28. The repository will be downloaded (it may spend several minutes "receiving objects").

29. Ensure that "Import existing projects" is selected and "Working Directory" is selected from the list; then click "Next"

30. All projects are selected by default; click "Finish"

31. Switch back to the Java perspective. The Package Explorer view should now contain several new projects with Photran's source code. There should be no compilation errors (although there will be a few warnings).

Last modified April 28, 2010

Part V. Running the test cases

32. In the Package Explorer view, select the `org.eclipse.photran.core.vpg.tests` project.

33. Right-click on that project and select Run As > Run Configurations.... A dialog will appear.

34. In that dialog, create a new **JUnit Plug-in Test** launch configuration. Call it "Photran-Tests".

35. For the configuration that you have just created, switch to the "Arguments" tab.

36. Change the "VM arguments" field to `-ea -Xms40m -Xmx512m`

37. Switch to the "Environment" tab.

38. *(Optional)* If you are running Linux or Mac OS X and have gfortran installed, some of Photran's refactoring unit tests can attempt to compile and run the Fortran test programs before and after the refactoring in order to ensure that the refactoring actually preserves behavior (and produces code that compiles). The following steps will enable this behavior. Note, however, that if the path to gfortran is incorrect, or if gfortran cannot be run successfully, it will cause the test suite to fail... so you might not want to do this the very first time you attempt to run the test suite.

    (a) Create a new environment variable called `COMPILER` with the full path to gfortran. This will be something like `/usr/local/bin/gfortran`

(b) Create a new environment variable called `EXECUTABLE` with a path to some non-existent file in your home directory, e.g., `/Users/joverbey/a.out`. When gfortran is run, it will write the executable to this path.

39. Click the "Run" button to run the tests. It will take at least a minute to run the test suite. When it finishes, you should get a green bar in the JUnit view. If you get a red bar, some of the tests failed; the JUnit view will have details.

40. To run the tests again later, just launch the "Photran-Tests" configuration from the Eclipse Run menu.

    Note. *UIUC personnel: See the appendix "Additional Information for UIUC Personnel" in the Photran Developer's Guide for information on additional unit test cases.*

Last modified May 19, 2010

Part VI. Launching Photran in the debugger

41. In the Package Explorer view, select the `org.eclipse.photran.core` project (or any other plug-in project).

42. Right-click on that project and select Debug As > Debug Configurations.... A dialog will appear.

43. In that dialog, create a new **Eclipse Application** launch configuration. Call it "Photran".

44. For the configuration that you have just created, switch to the "Arguments" tab.

45. Change the "VM arguments" field to:

    -ea -XX:PermSize=64m -XX:MaxPermSize=128m -Xms64m -Xmx768m

    (These arguments will enable assertions, increase the amount of PermGen space, and increate the amount of heap space available to Eclipse.)

46. *(Optional)* If you will be developing fixed form refactorings, or if you need fixed form refactoring enabled...

    (a) Switch to the "Environment" tab.

    (b) Create a new environment variable called `ENABLE_FIXED_FORM_REFACTORING` with a value of 1.

47. Click the "Debug" button. A new instance of Eclipse will open with the CDT and Photran plug-ins compiled from the code in your workspace.

48. To run it again later, just launch the "Photran" configuration from the Eclipse Run menu. Debug > Debug History > Photran will launch it in the debugger again (this will allow you to set breakpoints, watch expressions, etc.), while Run > Run History > Photran will launch it in a normal JVM (with debugging disabled).