Tycho-era development and build setup

Newer Doc

Newer setup documentation is available. Please refer to Eclipse development setup for CIShell and Sci2

This document is in the process of being deprecated. Updated documentation on the same topic may be found here: Eclipse development setup for CIShell and Sci2

Developer-facing

CIShell GitHub

- CIShell page at GitHub
    - Repository
    - Create a GitHub account
    - If you will be contributing changes back, create your own fork of the CIShell project.
        - And see this page for instructions on how to collaborate using GitHub
- Install EGit, an Eclipse Team provider for Git. To install, either:
    - Eclipse Marketplace > find: "EGit".
    - Install New Software > update site: http://download.eclipse.org/egit/updates
- See Git Setup for CIShell for help with Git. Also check out the Git Parable.

Maven Integration for Eclipse (a.k.a. m2eclipse or m2e)

Installing the required software

- http://eclipse.org/m2e/
- To install, either:
    - Eclipse Marketplace > find: "Maven Integration for Eclipse".
    - Install New Software > update site: http://download.eclipse.org/technology/m2e/releases
- Includes an embedded Maven installation.
- Tycho is a collection of Maven plugins. When Maven processes a POM using any of these plugins, they will be automatically downloaded and hooked in.
- When you're installing m2e connectors, you need to install both the **EGit** and the **Subclipse**. Run File->Import and select "Check out Maven Projects from SCM," then click the link to "Find more SCM connectors." Find theEGit connectorand the Subclipse connector (Eclipse's SVN Client) (for Sci2) and install them. Install all the required plugins, and also the CollabNet Merge Client and the SVNKit Client Adapter.
    - Some users may encounter a missing plugin error while instalilng the eGit connector (bundle org.eclipse.egit.core). This happens when Eclipse updates it's requirements and doesn't inform the Maven environment. A workaround is to go to the following URL: http://repository.tesla.io:8081/nexus/content/sites/m2e.extras/m2eclipse-egit/0.14.0/N/, click on the latest version, copy the URL and go to *Help > Install New Software...,* paste the URL and install the software. You should now have a working connector.
    - If this step does not work, try to open the repository in the "Git Repository Exploring" perspective. Select the projects you'd like to import and right click and select "Import Maven Projects...". If you do not see this option, install the m2e->egit connector.
    - If it is giving error - "Cannot complete the install because one or more required items could not be found." , it is mostly because of the Subclipse installation so first try to install EGit separately, it shouldn't give any error. Than go to Help > Install new software, Add the Eclipse update site URL for Subclipse: http://subclipse.tigris.org/update_1.6.x . Try to stick with Subclipse version 1.6 instead of version 1.8.
- After you've installed and restarted Eclipse, **go to Window > Preferences, type SVN in the filter box at the top left, click on the SVN node, and change the SVN interface to the SVNKit (PureJava) one**. Now SVN should work. (You can check it by going to File > Import > Maven > Check out Maven Projects from SCM, here you should see 'svn' option in drop down menu of SCM URL)

Checking out the CIShell code from the repo

- Run File->Import and select "Check out Maven Projects from SCM," give it the GitHub clone URL from your fork (if you will contribute), or the main repo (if you won't).
- CIShell is unexpectedly huge (maybe 700 megs?). Give it time.
- How to use remote repositories with EGit (note: since 90% of the instructions on the internet are for using git from the command line, I find it useful to also have a copy of command line git. Also, it doesn't seem to work if you generate a SSH keypair with a passphrase, using the git command line client: Eclipse doesn't seem to know how to ask for it. Bleh.)

Find the project called "parent".

- This is Eclipse's idea of the top-level directory of CIShell.
- Work around an m2e bug, and make Git realize that it's a git project: right-click parent and Team->Share project..., select Git, click the checkbox for "Use or create repository in parent folder of project." It should figure out that it's already a git project, so then click Finish. Now you can run git operations like committing and pulling from this project.

To build the entire CIShell project:

- Maven (step 1):
    - Right click "parent", then click Run as->Maven clean

        - After that finishes, right click "parent" again, then click Run as->Maven install. This may take some time
- Ant (step 2):
    - Some options cannot be configured by Maven during the build, so are done after the build using an Ant script
    - Before this will work, you will need to add the [Ant-Contrib](http://ant-contrib.sourceforge.net/) library to Ant.
        - To get the library, you can go to the aforementioned website, or simply download the latest version (as of 9/19/2013) here: ant-contrib-1.0b3-bin.zip
        - You do this by downloading the library, unzipping it, and placing the jar in ANT_INSTALLATION/lib/.
        - For me, the ANT_INSTALLATION/lib directory was eclipse/plugins/org.apache.ant_<version>/lib
        - Then, in Eclipse, you can add the jar to Ant's classpath from "Window -> Preferences -> Ant -> Runtime -> Classpath". Select Ant Home Entries and click Add External JARs, then navigate to the ant-contrib jar and select it.
    - Once that's done, in Eclipse, go to parent/deployment/org.cishell.reference.releng/postMavenTasks.xml. Right click and go to run as->ant build

Locating and launching the finished build:

- Go to eclipse/workspace/parent/deployment/org.cishell.reference.releng/target/products-final. There will be several compressed directories, each for a different system type. Locate the one for your system, and unzip it to a new directory
- Go into that directory, right click on the CIShell application, and create a shortcut. Right click on the shortcut and go to properties. At the end of the target line, after a space, type "-clean -console". This will aid with adding new plugins and debugging. Here is how the target line looks on my PC, for reference:
    - C:\Users\mvukas\Desktop\eclipse\workspace\parent\deployment\org.cishell.reference.releng\target\products-final\cishell-win32.win32.x86\cishell.exe -clean -console
- Launch the shortcut to run CIShell

Things that confuse Eclipse

- Maven projects are set up in nested directories, while Eclipse projects tend not to contain each other. This manifests in several ways:
    - If you have a complete SVN checkout and want to add one more plugin to it, you have to:
        1. Develop the plug-in
        2. Copy all the files from your developed plug-in into the source tree (and update pom.xml files and stuff)
        3. Delete the Eclipse project that refers to the old place your files were
        4. Import the project from the new place ("import existing maven projects")
    - EGit's diff viewer sometimes can't look at the difference between files that are in a nested project
- Eclipse and Maven have different ideas about which dependencies are fulfilled.
    - It's perfectly possible for Maven to be able to find all the dependencies of a project, while Eclipse can't, or vice versa.
        - Eclipse looks at the Target Platform, and any other projects in the workspace.
        - Maven will only look at other projects in the workspace if you run a build from the a directory that contains all the relevant projects. For instance, the root project of the checkout.
            - But, if you've already built the required projects with `mvn install` / Run as ->Maven install, Maven will have a copy of them in the local repository. This is good, except when you've just been fixing the required project and you want to test something that depends on it - have to remember to run Maven Install on both.

Build-facing

Libraries

Prefer re-use of existing, externally-hosted library bundles over the current practice of creating thin CIShell wrappers around locally-hosted jars.

Resources:

- Eclipse Orbit.
- Repositories specific to individual organizations, projects, or libraries.

Jenkins jobs for builds of CIShell etc.

http://cns-ts1.slis.indiana.edu:8080/jenkins/

Command-line instructions for building CIShell and Sci2

http://in.cns.iu.edu/svn/nwb/branches/tycho/README.md

How to publish features and bundles to a p2 repository

Eclipse has a built-in application for this that you can invoke at the command line. `SRC` should be a directory with subdirectories named `plugins` and `features`. You can use scripts similar to the following.

- Windows

publish-p2.bat

```
set SRC=C:/source
set DEST=C:/repository

eclipse.exe -debug -consolelog -nosplash -verbose \
  -application org.eclipse.equinox.p2.publisher.FeaturesAndBundlesPublisher \
  -metadataRepository file:%DEST% \
  -artifactRepository file:%DEST% \
  -source %SRC% \
  -compress -append -publishArtifacts
```

- Linux

publish-p2.sh

```
#!/bin/bash
src=/path/to/source
dest=/path/to/repository
rm -rf "$dest"
dest="file:$dest"

/path/to/eclipse -nosplash \
    -application org.eclipse.equinox.p2.publisher.FeaturesAndBundlesPublisher \
    -metadataRepository "$dest" \
    -artifactRepository "$dest" \
    -repositoryName "Repository Name" \
    -source "$src" \
    -compress -append -publishArtifacts
```